Systementwurf für verteilte Applikationen und Modelle im Bauplanungsprozess

Daniel G. Beer

Dissertation zur Erlangung des akademischen Grades

Doktor - Ingenieur

an der Fakultät Bauingenieurwesen der Bauhaus-Universität Weimar

Gutachter:

1. Prof. Dr.-Ing. Karl E. Beucke, Bauhaus-Universität Weimar
2. Prof. Dr.-Ing. habil. Dr.-Ing. E.h. Udo F. Meißner, TU Darmstadt
3. Prof. Dr.-Ing. Stefan M. Holzer, Universität der Bundeswehr München
4. Prof. Dr.-Ing. Berthold Firmenich, Bauhaus-Universität Weimar

Eingereicht am: 21.12.2005

Tag der Disputation: 30.03.2006

Vorwort

Die vorliegende Dissertation entstand in der Zeit von Januar 2003 bis Dezember 2005 während meiner Tätigkeit als wissenschaftlicher Mitarbeiter an der Bauhaus-Universität Weimar am Lehrstuhl „Informatik im Bauwesen“ bei der Bearbeitung des von der Deutschen Forschungsgemeinschaft DFG geförderten Forschungsprojekts „interCAD“[1] im Schwerpunktprogramm 1103[2].

Während meiner von Januar 2001 bis Dezember 2002 dauernden Mitarbeit im ebenfalls von der DFG geförderten Sonderforschungsbereich 524[3] im Projekt D1[4,5] des Teilbereiches D[6] erwarb ich die dafür erforderliche Methodik, auch wenn die thematische Ausrichtung eine andere war.

Zum Gelingen der Arbeit haben viele Personen beigetragen. Ihnen fühle ich mich zu großem Dank verpflichtet.

Mein besonderer Dank gilt Herrn Professor Beucke, der als mein Mentor das nötige Umfeld und Freiräume für die Bearbeitung meiner Dissertation geschaffen hat. Er gab mir wertvolle Hinweise und persönlichen Rat. Auch in organisatorischer Hinsicht hat er mir unkompliziert viele Steine aus dem Weg geräumt.

Herrn Professor Meißner und Herrn Professor Holzer danke ich für die unkomplizierte Übernahme des Gutachtens. Den fachlichen Rat von Professor Meißner habe ich als Bearbeiter im DFG-Schwerpunktprogramm 1103, zu dessen Koordinatoren er seit Beginn an gehört, sowie auch in anderen gemeinsamen Projekten schätzen gelernt.

Herrn Professor Firmenich gilt mein Dank in persönlicher und fachlicher Hinsicht. Ich habe seine Kompetenz und Persönlichkeit während meiner Arbeit am Lehrstuhl sehr schätzen und würdigen gelernt. Durch seine praktischen Erfahrungen und wissenschaftliche Kompetenz hat er meiner Arbeit wesentliche Impulse verliehen.

Ich möchte meiner Familie – meiner Großmutter Helene Prager†, meiner Mutter Hildegard, meinem Vater Günther – und meiner Freundin Enia ganz herzlich danken, dass sie mich in der langen und entbehrungsreichen Zeit unterstützt und immer wieder aufgerichtet haben. Ohne sie wäre diese Arbeit sicher nicht möglich gewesen.

Insbesondere danke ich meinen Kolleginnen und Kollegen am Lehrstuhl „Informatik im Bauwesen“ für die konstruktive Zusammenarbeit und die Unterstützung in fachlichen, technischen, organisatorischen und nicht zuletzt menschlichen Aspekten. Besonders erwähnen möchte ich hier meinen Freund und Mitarbeiter im Projekt Torsten Richter, die Mitdoktoranden Christian Koch, Timo Heinrich, Bertie Olivier und Mohamed Nour sowie Jens-Uwe Wagner und Mechthild Bieber, ohne die ich eine so stabile und funktionierende Informationsinfrastruktur sicher nicht vorgefunden hätte. Zum Abschluss danke ich ganz besonders Frau Roswitha Eckart für die Korrekturen an Ausdruck, Rechtschreibung und Grammatik.

Weimar, den 31. März 2006 *Daniel G. Beer*

[1] „Entwurf und Verifizierung einer CAD-Systemarchitektur zur Unterstützung der verteilten technischen Bearbeitung im Konstruktiven Ingenieurbau“ [Firmenich u. a. 2004]
[2] „Vernetzt-kooperative Planungsprozesse im Konstruktiven Ingenieurbau“ [Meißner u. Rüppel 1999]
[3] „Werkstoffe und Konstruktionen für die Revitalisierung von Bauwerken“ [Bucher u. Müller 2004]
[4] Phase 1: „Strukturierung von Informationen und Prozessen für Revitalisierungsaufgaben“ [Beucke u. a. 2002]
[5] Phase 2: „Methoden und Werkzeuge zur Planung der Bauplanung“ [Beucke u. a. 2004a]
[6] „Informationsverarbeitung und Kommunikation“ [Beucke u. a. 2004b]

Für

Helene, Hildegard, Günther

&

Enia

Kurzfassung

Der Planungsprozess im Konstruktiven Ingenieurbau ist gekennzeichnet durch drei sich zyklisch wiederholende Phasen: die Phase der Aufgabenverteilung, die Phase der parallelen Bearbeitung mit entsprechenden Abstimmungen und die Phase der Zusammenführung der Ergebnisse. Die verfügbare Planungssoftware unterstützt überwiegend nur die Bearbeitung in der zweiten Phase und den Austausch der Datenbestände durch Dokumente.

Gegenstand der Arbeit ist die Entwicklung einer Systemarchitektur, die in ihrem Grundsatz alle Phasen der verteilten Bearbeitung und unterschiedliche Arten der Kooperation (asynchron, parallel, wechselseitig) berücksichtigt und bestehende Anwendungen integriert. Das gemeinsame Arbeitsmaterial der Beteiligten wird nicht als Dokumentmenge, sondern als Menge von Objekt- und Elementversionen und deren Beziehungen abstrahiert. Elemente erweitern Objekte um applikationsunabhängige Eigenschaften (Features). Für die Bearbeitung einer Aufgabe werden Teilmengen auf Basis der Features gebildet, für deren Elemente neue Versionen abgeleitet und in einen privaten Arbeitsbereich geladen werden. Die Bearbeitung wird auf Operationen zurückgeführt, mit denen das gemeinsame Arbeitsmaterial konsistent zu halten ist.

Die Systemarchitektur wird formal mit Mitteln der Mathematik beschrieben, verfügbare Technologie beschrieben und deren Einsatz in einem Umsetzungskonzept dargestellt. Das Umsetzungskonzept wird pilothaft implementiert. Dies erfolgt in der Umgebung des Internet in der Sprache JAVA unter Verwendung eines Versionsverwaltungswerkzeuges und relationalen Datenbanken.

Abstract

The planning process in structural engineering can be characterized by three iterative phases: the phase of distribution of tasks, the phase of parallel working with cooperation among the planners and the phase of merging the results. Available planning software does only support the second phase and the exchange of data via documents.

The objective of this thesis is the development of a software architecture that supports the three phases and all types of cooperation (asynchronous, parallel and reciprocal) in principle and integrates existing engineering applications. The common planning material is abstracted as a set of object versions, element versions and their relationships. Elements extend objects with application independent properties, called features. Subsets on the base of features are calculated for the execution of tasks. Therefore new versions of elements and objects are derived and copied into the planners's private workspace. Already stored versions remain unchanged and can be referred to. Modifications base on operations that ensure the consistency of the versioned model.

The system architecture is formally described with mathematical methods. Available information technology is analyzed and used for an implementation concept. The implementation concept is proven by a pilot applicable in the Internet. The implementation is based on the programming language JAVA, a version control system and a relational database.

Kurzinhalt

Inhaltsverzeichnis

Kapitel 1

Einleitung

„*Most application software for civil engineering purposes [...] was originally designed as a stand-alone application [...] at a time when neither the necessary requirements for computer and network technology nor for physically available wide-area network connections were fulfilled. [...] With the availability of an affordable and ubiquitous network environment [...], the network developed into a vital part of engineering work. Every engineer today utilizes network functionality for communication purposes, however comparatively few engineers use it effectively for cooperation purposes. The concepts and software needed for these purposes are simply not in a state that is widely acceptable to the civil engineering user community.*“

aus [Beucke u. Beer 2005]

„*Das DFG-Schwerpunktprogramm 'Vernetzt-kooperative Planungsprozesse im Konstruktiven Ingenieurbau' hat [...] zum Ziel, die Planungsprozesse des Konstruktiven Ingenieurbaus für die Nutzung verteilter Ressourcen zu gestalten, geeignete Kooperationsmodelle für die Fachplanung im Informationsverbund der Projektbeteiligten zu entwickeln und die kooperative Projektbearbeitung unter Nutzung verteilter Fachmodelle in Netzen zu ermöglichen. [...] Von zentralem Interesse ist [...] die Verbesserung der Qualität und Effizienz der verteilten Ingenieurplanung [...]. Insbesondere sind neue Grundlagen für verteilte Ingenieurmodelle und kooperative Planungsprozesse [...] zu entwickeln.*“

aus [Meißner u. Rüppel 1999]

„*Die Planer kommunizieren miteinander durch den Austausch von [...] Dokumenten. [...] Die Dokumente verschiedener [...] Systeme sind in der Regel unverträglich. Neben dem Verlust von Informationen bei der Datenübertragung ist es für den Empfänger [...] schwierig, die Änderungen visuell zu beurteilen und entsprechend zu reagieren. Da kein gemeinsames logisches Planungsmaterial für den Gesamtprozess vorhanden ist, können Beziehungen zwischen den Objekten nicht beschrieben werden. Weil Beziehungen in der Form von Bindungen zwischen den Objekten fehlen, werden Änderungen nicht korrekt gehandhabt und das gemeinsame Planungsmaterial ist fast immer inkonsistent.* “

aus [Firmenich 2001]

1.1 Problemstellung

Bauplanungsprozess: Projekte des *Bauplanungsprozesses*[1] werden durch viele unterschiedliche Akteure bearbeitet. Sie erzeugen gemeinsame Bauwerksmodelle[2] unter Verwendung von Methoden fachspezifischer Planungswerkzeuge. Prozessmodelle strukturieren den Ablauf des Planungsprozesses und beschreiben die Erstellung der Bauwerksmodelle[3]. Sie bestehen aus Zuständen, die mit Teilen der Bauwerksmodelle verbunden sind, und Aktivitäten, die Methoden zur Bearbeitung der Bauwerksmodelle verwenden. Den Akteuren im Planungsprozess sind Rollen zugeordnet, die sie für Aktivitäten befähigen oder von diesen ausschließen. Zwischen den Akteuren bestehen vielfältige Beziehungen, die mit Hilfe von Organisationsmodellen beschrieben werden. Die Akteure nutzen Kommunikationsmittel (Telefon, E-Mail usw.), um sich untereinander abzustimmen.

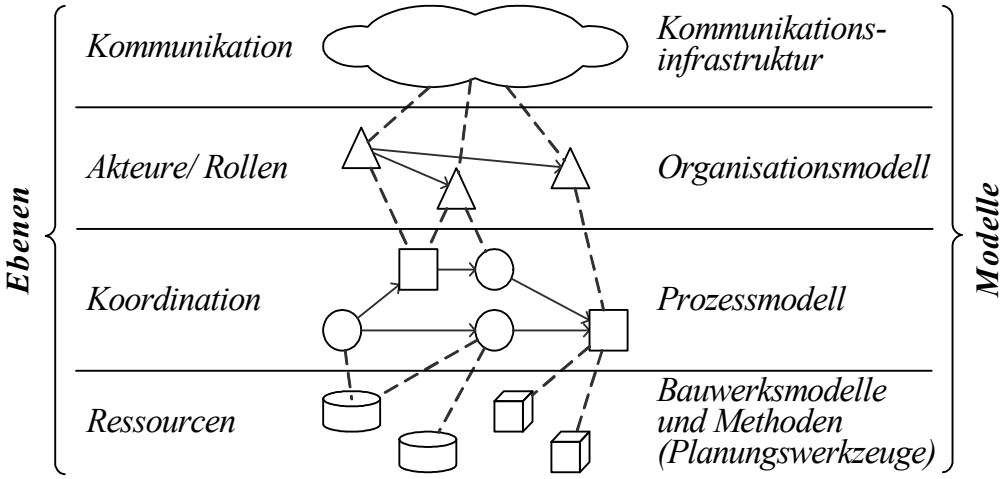


Abbildung 1.1: Kooperationsmodell einer vernetzt-kooperativen Planung im Konstruktiven Ingenieurbau nach [Meißner u. Rüppel 1999]

Auf Prozessebene sind drei sich zyklisch wiederholende Phasen zu erkennen. In Phase (1) bilden die Akteure Teilmengen des gemeinsamen Planungsmaterials für die eigene Bearbeitung. In Phase (2) werden diese Teilmengen synchron oder asynchron bearbeitet. In Phase (3) schließlich werden die Ergebnisse der Bearbeitung im gemeinsamen Planungsmaterial gespeichert.

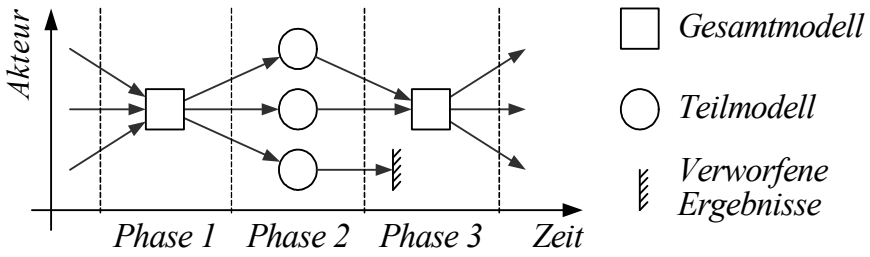


Abbildung 1.2: Phasen der kooperativen Bearbeitung in Anlehnung an [Firmenich 2001]

[1] [Meißner u. a. 2000], [Rüppel u. Meißner 2000], [Meißner u. Rüppel 2003], [Rüppel u. Meißner 2003]
[2] [Firmenich 2004a], [Firmenich 2004d]
[3] [Beucke u. a. 2002], [Beucke u. a. 2004a], [Klauer 2005]

Kooperation: Der Bauplanungsprozess ist durch seinen iterativen Charakter gekennzeichnet. Die endgültige Lösung wird als Folge der Abstimmungsprozesse erst nach mehrmaligen Änderungen der Lösungen der einzelnen Fachdisziplinen erzielt. [Bretschneider 1998] beschreibt unterschiedliche Arten der *Kooperation* in Abhängigkeit des bearbeiteten Planungsmaterials und der Zeit. Überschneidet sich die Bearbeitungszeit, so spricht man von synchroner, andernfalls von asynchroner Kooperation. Die synchrone Kooperation wird noch einmal unterteilt: Überschneidet sich das bearbeitete Planungsmaterial, so liegt wechselseitige, ansonsten parallele Kooperation vor. In der derzeitigen Praxis kooperieren die Akteure durch den Austausch von Dokumenten, wodurch nicht alle Kooperationsarten möglich sind.

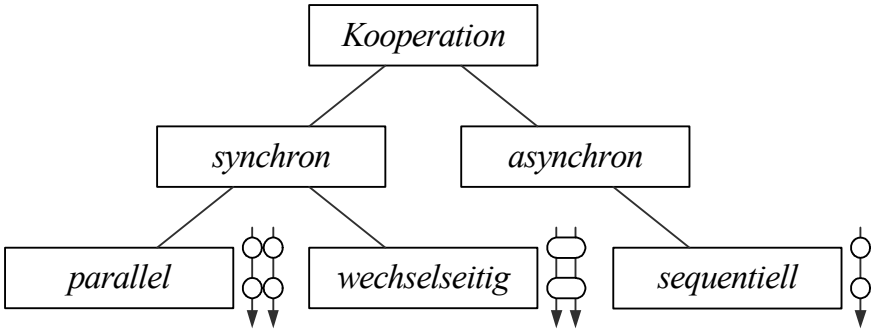


Abbildung 1.3: Kooperation nach Zeit und Gegenstand der Bearbeitung nach [Bretschneider 1998]

Dokument: Die Struktur eines *Dokuments* wird von der sie erzeugenden Ingenieuranwendung vorgegeben und ist im Allgemeinen nur dieser bekannt. Beim Austausch von Dokumenten zwischen unterschiedlichen Ingenieuranwendungen ist eine Transformation erforderlich, bei der in der Regel Informationen verloren gehen, was durch mehrmalige Transformation sogar kumuliert[4]. Standards für Bauwerksmodelle[5] lösen dieses Problem nur teilweise, da die Fachapplikationen optimierte Objektmodelle verwenden, die nicht den gleichen Modellumfang haben.

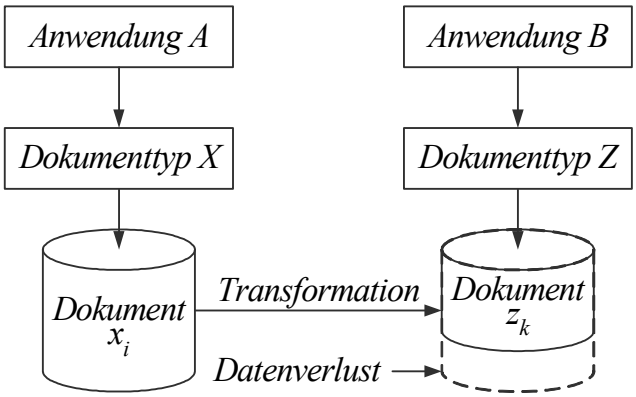


Abbildung 1.4: Datenverlust beim Dokumentenaustausch

[4] [Firmenich 2004c], [Firmenich 2004b], [Koch 2005]
[5] Siehe zum Beispiel [IAI 2004].

Beziehungen: Zwischen Objekten in Dokumenten, die von unterschiedlichen Ingenieuranwendungen erzeugt werden, können im Allgemeinen keine *Beziehungen* gebildet werden[6]. Damit lassen sich logische Abhängigkeiten, die vielfach zwischen Teilen der Dokumente bestehen, nicht abbilden. [Pahl u. Beucke 2000] nennt diese Beziehungen Bindungen.

Redundanz: Akteure unterschiedlicher Fachdisziplinen benötigen oder bearbeiten gleiche Informationen, zum Beispiel die Bauwerksgeometrie, wenn auch in unterschiedlicher Ausprägung und Genauigkeit. Sie verwenden dafür verschiedene Ingenieuranwendungen und erzeugen unterschiedlich strukturierte Dokumente. Damit liegen gleiche Informationen in verschiedenen Dokumenten mehrfach – *redundant* – und unterschiedlich strukturiert vor. Zwischen diesen gleichen Informationen können jedoch im Allgemeinen keine Beziehungen gebildet werden.

Konsistenz: Bei Änderungen in einem Dokument können die Auswirkungen auf andere Dokumente nicht automatisch ermittelt werden, weil zwischen diesen keine Beziehungen bestehen. Die *Konsistenz* des gemeinsamen Planungsmaterials kann nicht sichergestellt werden[6].

Nachvollziehbarkeit: Die rechtliche und fachliche *Nachvollziehbarkeit* von Änderungen ist nur dann gewährleistet, wenn durch Änderungen Dokumente nicht überschrieben werden, sondern neue Versionen angelegt werden. Aber auch dann ist eine Nachvollziehbarkeit erschwert, da auf Grund unterschiedlicher Strukturen und fehlender Beziehungen eine Vergleichbarkeit unterschiedlicher Dokumente nur in wenigen Fällen möglich ist.

Varianten: Im Bauplanungsprozess ist es üblich, unterschiedliche Lösungen zu entwerfen, diese gegenüber zu stellen und ausgewählte *Varianten* weiter zu verfolgen. Zur Abbildung dieser Planungsentscheidungen ist eine lineare Versionierung (Historie), wie sie zum Beispiel von Dokumentmanagementsystemen verwendet wird, nicht ausreichend, da Verzweigungen und Zusammenführungen nicht erkennbar sind.

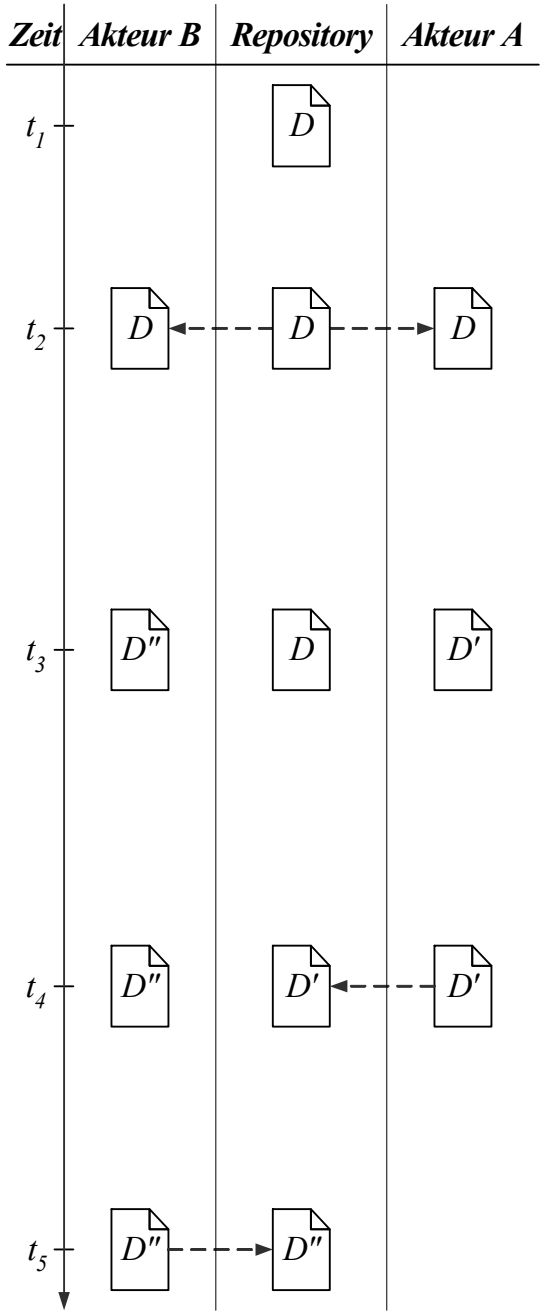
1.2 Szenarien

Szenarien: Im folgenden werden verschiedene *Szenarien* für die verteilte Bearbeitung vorgestellt und dabei unterschiedliche Konzepte untersucht. Es wird der Zustand des lokalen und gemeinsamen Planungsmaterials zu verschiedenen Zeitpunkten dargestellt. Grundlage ist ein gemeinsamer Datenspeicher – das Repository (*engl. „Ablage, Speicher“*) – über den die Akteure die Daten austauschen. Diese werden in den Szenarien in Dokumenten, Datenbanken oder Objektmengen strukturiert.

Schlussfolgerungen: Es wird untersucht, inwieweit die beschriebenen Probleme mit den verschiedenen Ansätzen gelöst werden können. Das betrifft vor allem die Arten der Kooperation, die Variantenbildung, die Beziehungen innerhalb des Planungsmaterials, die Redundanz und Konsistenz sowie die Nachvollziehbarkeit. Die *Schlussfolgerungen* aus den Szenarien dienen dazu, ein für den Planungsprozess geeignetes Konzept zur Unterstützung aller Kooperationsarten nach [Bretschneider 1998] zu finden.

[6] [Firmenich 2001]

1.2.1 Der Letzte gewinnt



Zeitpunkt 1: Im Repository ist ein Dokument D vorhanden, das synchron von zwei Bearbeitern A und B bearbeitet werden soll. Beide haben Zugriff auf das Dokument und arbeiten unabhängig voneinander.

Zeitpunkt 2: Bearbeiter A und B holen sich gleichzeitig das Dokument und öffnen es. Es gibt jetzt drei identische Dokumente: eines im Repository und zwei Kopien bei den Bearbeitern.

Zeitpunkt 3: Beide Akteure bearbeiten ihr Dokument wechselseitig und unabhängig voneinander. Änderungen eines Bearbeiters werden dem anderen Bearbeiter nicht mitgeteilt und sie können auch nicht automatisch ermittelt werden, da keine Beziehungen zwischen den Dokumenten bestehen. Bearbeiter A erstellt Dokument D', Bearbeiter B erstellt Dokument D''.

Zeitpunkt 4: Bearbeiter A hat seine Änderungen fertig gestellt und speichert das Dokument D' im Repository. Dabei wird das Dokument D überschrieben. Bearbeiter B wird darüber nicht informiert.

Zeitpunkt 5: Nun hat Bearbeiter B seine Änderungen fertig gestellt und speichert das Dokument D'' im Repository. Dabei wird das Dokument D' und damit die Fassung von A überschrieben. Als Ergebnis bleibt die von B erzeugte Datei D'' im Repository gespeichert.

Schlussfolgerungen: Die wechselseitige Bearbeitung von Dokumenten nach dem Prinzip „der Letzte gewinnt“ führt zum Verlust von Daten. Nur die Ergebnisse des zuletzt speichernden Bearbeiters sind verfügbar.

Um Datenverlust zu verhindern, ist die wechselseitige Bearbeitung gleicher Dokumente auszuschließen und eine parallele Bearbeitung zu erzwingen. Das ist nach diesem Ansatz zwar möglich, muss jedoch von den Beteiligten vereinbart und selbst überwacht werden.

Dies wird durch das Konzept der Sperre systematisch unterstützt.

1.2.2 Sperren

Zeitpunkt 1: Im Repository ist ein Dokument D vorhanden, das nicht synchron von mehreren Bearbeitern bearbeitet werden darf. Das wird durch eine Zugriffssperre auf dem Dokument erreicht.

Zeitpunkt 2: Bearbeiter A holt sich das Dokument D und öffnet es. Es gibt jetzt zwei identische Dokumente: eines im Repository und eine Kopie beim Bearbeiter A. Für weitere Bearbeiter ist das Dokument D gesperrt, es kann nur gelesen werden.

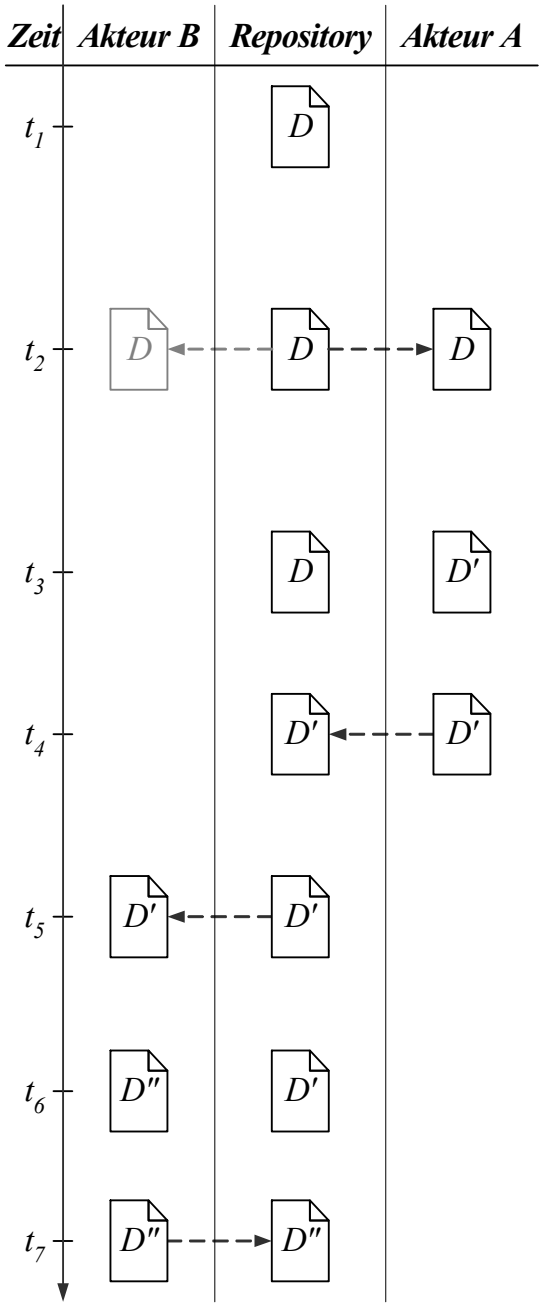
Zeitpunkt 3: Akteur A bearbeitet das Dokument D exklusiv. Er erstellt die Version D'.

Zeitpunkt 4: Bearbeiter A speichert D' im Repository. Dabei wird das Dokument D überschrieben. Er hebt die Sperre auf dem Dokument auf.

Zeitpunkt 5: Nun kann Bearbeiter B das von A erstellte Dokument D' aus dem Repository laden.

Zeitpunkt 6: Akteur B bearbeitet das Dokument D' exklusiv. Er erstellt die Version D''.

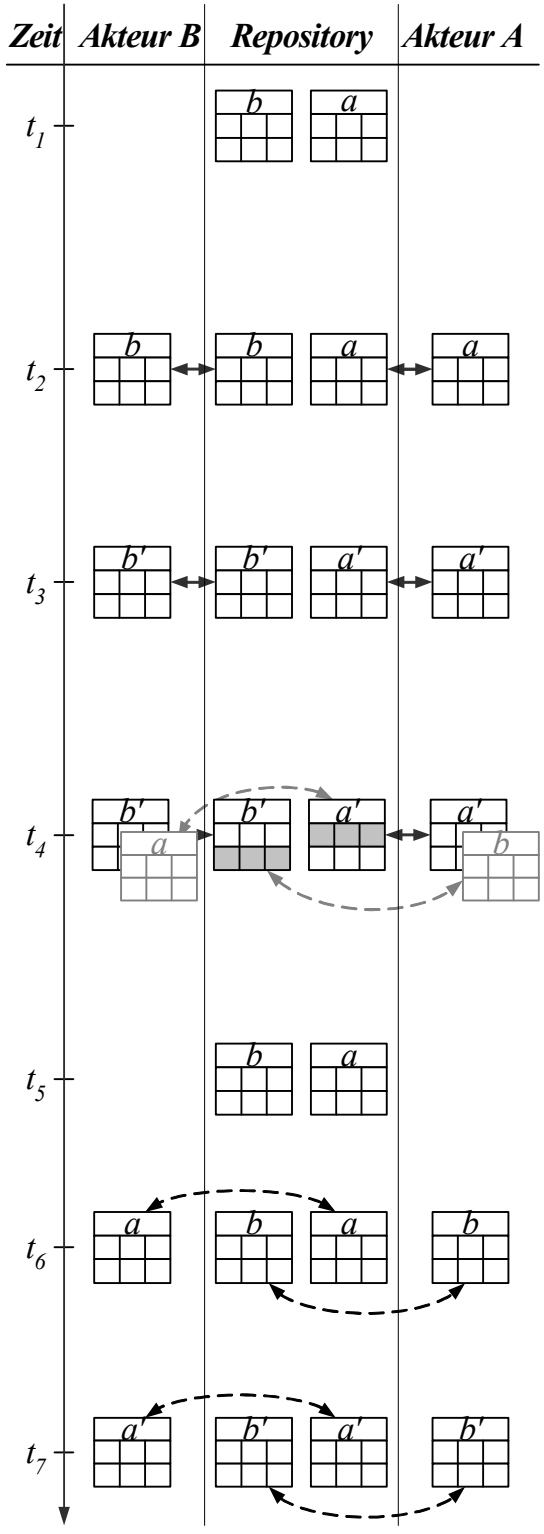
Zeitpunkt 7: Bearbeiter B speichert D'' im Repository. Dabei wird das Dokument D' überschrieben. Er hebt die Sperre für eine weitere Bearbeitung auf.



Schlussfolgerungen: Sperren verhindern das wechselseitige Arbeiten an gleichen Dokumenten durch Sequentialisierung. Unterschiedliche Dokumente können jedoch parallel verändert werden. Die Konsistenz kann durch Sperren nicht garantiert werden, wenn logische Beziehungen, die zwischen Dokumenten bestehen, bei der Bearbeitung durch mehrere Akteure nicht beachtet werden. Dies ist im Allgemeinen auch nur schwer möglich, da Beziehungen nicht explizit vorliegen und die Konsistenz nicht automatisch geprüft werden kann.

Ist die gesperrte Datenmenge sehr groß, zum Beispiel bei Verwendung großer Dokumente, so ist es wahrscheinlich, dass Akteure warten müssen. Das Problem wird durch Sperren, die versehentlich nicht wieder aufgehoben wurden, verstärkt. Eine Verkürzung der Bearbeitungszeit umgeht dieses Problem. Eine Umsetzung sind kurze Transaktionen in Datenbanken.

1.2.3 Kurze Transaktionen



Zeitpunkt 1: Im Repository sind zwei Tabellen oder Datensätze a und b einer Datenbank vorhanden, die synchron von zwei Bearbeitern A und B bearbeitet werden sollen. Eine Sperre verhindert die wechselseitige Bearbeitung eines Datensatzes.

Zeitpunkt 2: Bearbeiter A und B holen sich eine Verbindung für die Tabelle bzw. Datensätze a bzw. b. Es beginnt jeweils eine Transaktion, die beide Tabellen bzw. Datensätze für andere Bearbeiter sperrt.

Zeitpunkt 3: Beide Bearbeiter verändern die von ihnen geöffneten Daten a und b. Die Auswirkungen werden nach einem commit (*dt. „anvertrauen“*) *sofort* im Repository sichtbar.

Zeitpunkt 4: Bearbeiter A benötigt schreibenden Zugriff auf Teile des Datensatzes b, um seine Änderungen an a fertigzustellen. Benötigt nun Bearbeiter B schreibenden Zugriff auf Teile des Datensatzes a, um seine Änderungen an b zu vollenden, so entsteht eine Verklemmung (*engl. „dead lock“*) infolge der Sperren.

Zeitpunkt 5: Beide Bearbeiter brechen ihre Transaktionen ab und stellen den vorherigen Zustand a bzw. b wieder her (*engl. „roll back“*).

Zeitpunkt 6: Die Bearbeiter stellen nun eine Verbindung zu der jeweils anderen Tabelle bzw. dem anderen Datensatz her.

Zeitpunkt 7: Beide ändern die Daten mit sofortiger Auswirkung auf das Repository. Nun kann eine Bearbeitung der ursprünglich gewünschten Daten a bzw. b durch A bzw. B erfolgen.

Schlussfolgerungen: Kurze Transaktionen ermöglichen quasi wechselseitiges Arbeiten, da Änderungen sofort in das Repository übernommen werden. Es können jedoch Verklemmungen auftreten. Außerdem ist eine dauerhafte Netzwerkverbindung erforderlich, was unabhängiges, lokales Arbeiten ausschließt und nicht der Arbeitsweise im Bauplanungsprozess entspricht.

1.2.4 Versionshistorie

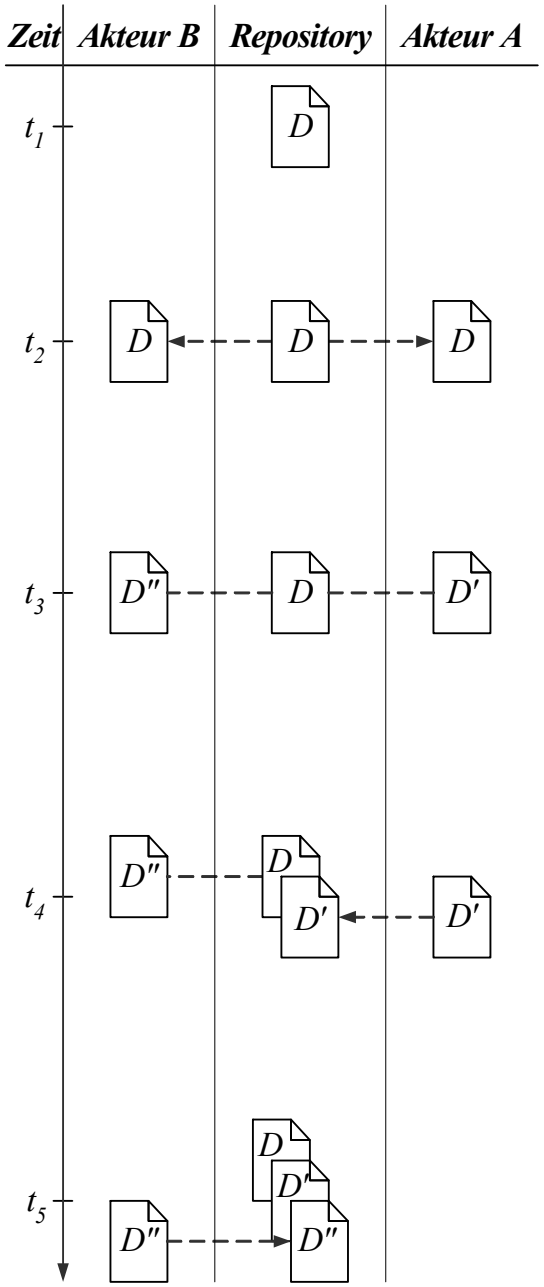
Zeitpunkt 1: Ein Dokumentenmanagementsystem[7] verwaltet ein Dokument D des Repository. Es kann verschiedene Zustände (Versionen) ein und des selben Dokuments und dessen Entstehungsgeschichte (Versionshistorie) speichern.

Zeitpunkt 2: Bearbeiter A und B laden sich das Dokument D. Sie erhalten eine *Kopie* für die Bearbeitung.

Zeitpunkt 3: Beide Akteure ändern ihre Kopie des Dokuments unabhängig und zeitgleich. Das Dokumentenmanagementsystem kann jedoch immer eine logische Beziehung zwischen ursprünglichem Dokument und (geänderter) Kopie durch Metadaten (Identifikatoren, Versionsnummern usw.) herstellen.

Zeitpunkt 4: Bearbeiter A speichert seine Änderungen als Dokumentversion D'. Die ursprüngliche Version D wird *nicht* überschrieben, sondern bleibt erhalten. Das Dokumentenmanagementsystem speichert die Historie der gespeicherten Dokumente.

Zeitpunkt 5: Bearbeiter B speichert seine Änderungen als Dokumentversion D''. Die vorhandenen Versionen D und D' bleiben erhalten. Bearbeiter B hat jedoch auf nicht mehr aktuellen Daten des Dokuments D gearbeitet, während neuere Daten mit dem Dokument D' durch den Bearbeiter A schon vorlagen.

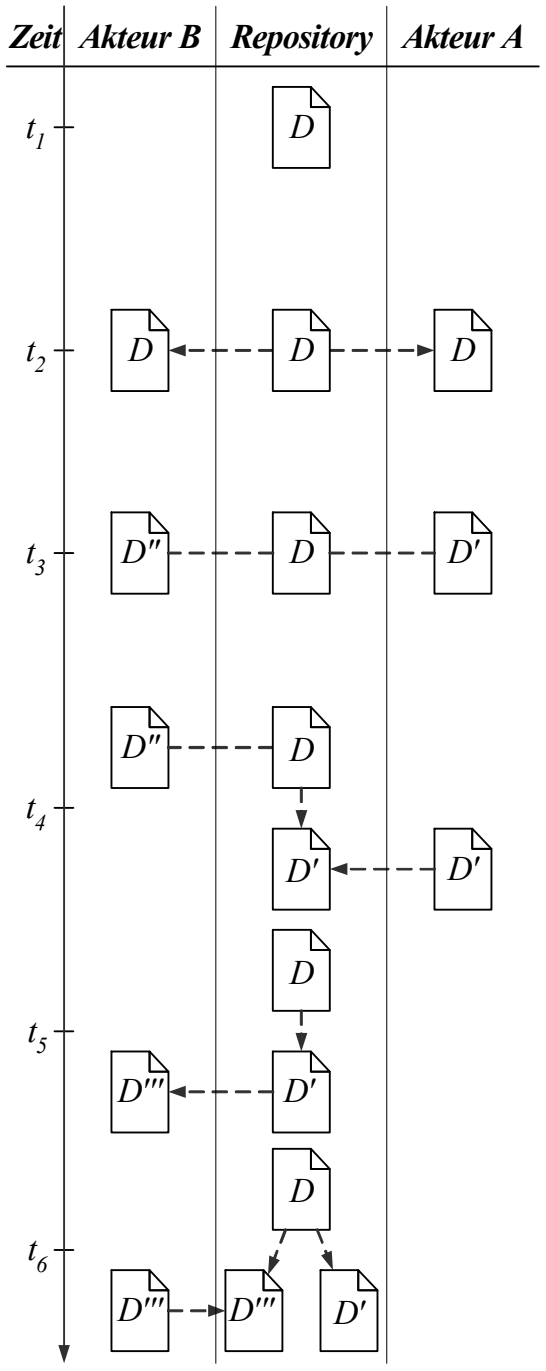


Schlussfolgerungen: Dokumentenmanagementsysteme und auch Internetportale mit dieser Funktionalität ermöglichen paralleles Arbeiten. Sie unterstützen auch die Vervielfältigung und Versionierung von Dokumenten in einer linearen Historie. Dadurch werden quasi lange Transaktionen unterstützt.

Die Historie gibt jedoch keine Auskunft darüber, ob eine Version eine Revision, also eine Überarbeitung, oder eine Variante zu einer anderen Version ist. Damit ist auch eine Zusammenführung von Varianten erschwert. Um solche Unterscheidungen treffen zu können, müssen Versionsverwaltungssysteme verwendet werden.

[7] [Klingelhöller 2001], [Müller u. a. 2003]

1.2.5 Versionierung



Zeitpunkt 1: Ein Versionsverwaltungssystem verwaltet ein Dokument D des Repository. Es kann verschiedene Versionen eines Dokuments und dessen Entstehungsgeschichte aus Revisionen, Verzweigungen, Varianten und Zusammenführungen (Versionsgraph) speichern.

Zeitpunkt 2: Bearbeiter A und B laden sich das Dokument D. Sie erhalten eine *Kopie* für die Bearbeitung.

Zeitpunkt 3: Beide Akteure ändern ihre Kopie des Dokuments unabhängig. Das Versionsverwaltungssystem kann jedoch immer eine logische Beziehung zwischen ursprünglichem Dokument und (geänderter) Kopie herstellen.

Zeitpunkt 4: Bearbeiter A speichert seine Änderungen als Dokumentversion D'. Die ursprüngliche Version D wird *nicht* überschrieben. Im Versionsgraphen wird die Bearbeitungsgeschichte des Dokuments gespeichert.

Zeitpunkt 5: Bearbeiter B *aktualisiert* seine Version D'' teilweise mit Ergebnissen D' des Bearbeiters A zur Version D'''.

Zeitpunkt 6: Bearbeiter B nimmt keine weiteren Änderungen vor und speichert die Version D''' als *Variante*. Das Versionsverwaltungssystem speichert diesen Zusammenhang im Versionsgraphen, so dass deren Zusammenführung möglich wird.

Schlussfolgerungen: Versionsverwaltungssysteme ermöglichen quasi wechselseitiges Arbeiten, da jederzeit ein Abgleich mit den Versionen des Repository und einer Verschmelzung der Ergebnisse möglich ist.

Der Ansatz ist jedoch nur im Umgang mit Textdokumenten anwendbar, da für den Abgleich und die Zusammenführung dem Versionsverwaltungssystem der Inhalt eines Dokuments bekannt sein muss. Bei Textdokumenten ist das der Fall, es werden jeweils Zeilen und deren Zeichen miteinander verglichen. Ein semantischer Vergleich ist jedoch nicht möglich.

1.2.6 Objektversionierung

Zeitpunkt 1: Ein Objektversionsverwaltungssystem verwaltet die Versionen a_1 und b_1 der Objekte a und b des Repository. Es kann den Versionsgraphen der Objekte speichern. Zusätzlich ist eine Beziehung (a_1, b_1) zwischen den Objektversionen modelliert.

Zeitpunkt 2: Bearbeiter A lädt Objektversion a_1, Bearbeiter B möchte Objektversion b_1 bearbeiten. Beide bekommen eine neue Objektversion als Kopie des Originals. b_2 bleibt weiter an a_1 gebunden.

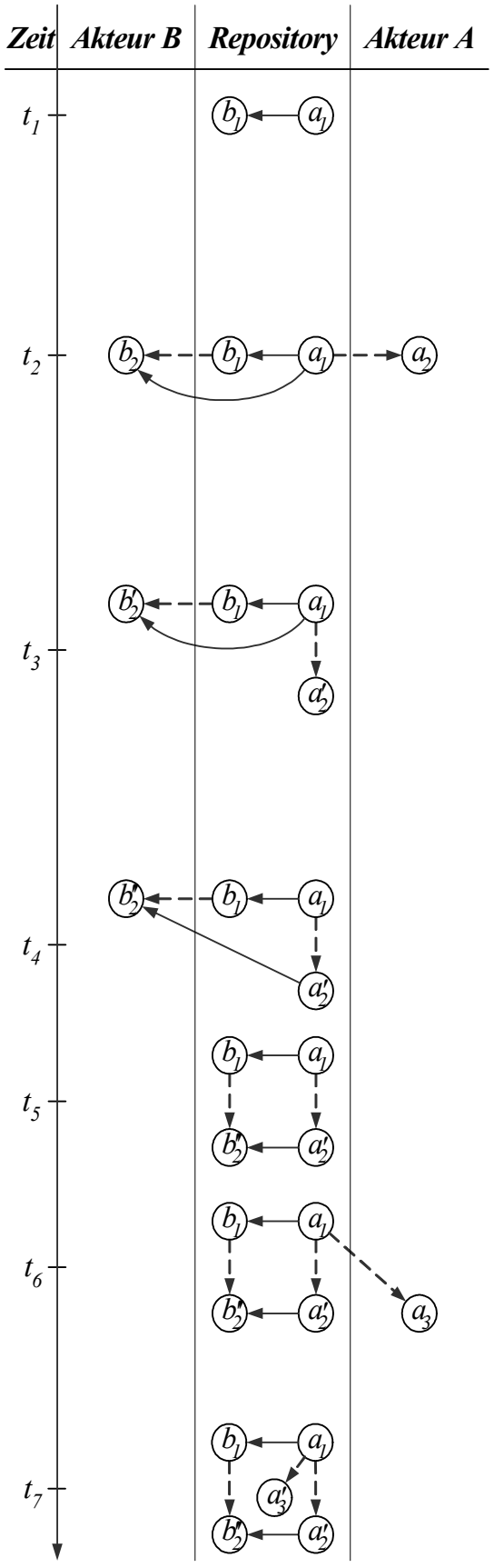
Zeitpunkt 3: Beide Akteure bearbeiten ihre Objektversionen unabhängig. Diese können im Gegensatz zu im Repository gespeicherten Objektversionen verändert werden. Zur Verdeutlichung werden lokale Änderungen durch hochgestellte Striche angedeutet, die jedoch nicht die Identität verändern: A ändert a_1 zu $a_2^{'}$ und speichert diese Version. Im Versionsgraphen wird die Bearbeitungsgeschichte $(a_1, a_2^{'})$ gespeichert.

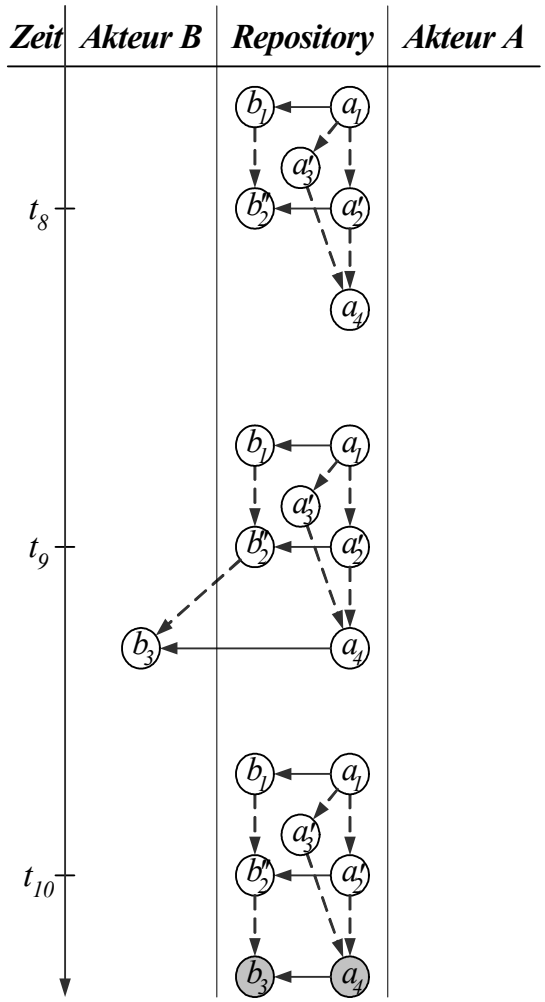
Zeitpunkt 4: Bearbeiter B aktualisiert seine Version $b_2^{'}$ mit dem Repository und erzeugt so Version $b_2^{''}$, die jetzt von $a_2^{'}$ abhängt.

Zeitpunkt 5: Er speichert seine aktualisierte Version $b_2^{''}$ im Repository. Die Bearbeitung wird als Paar $(b_1, b_2^{''})$ gesichert.

Zeitpunkt 6: Bearbeiter A möchte eine Variante zu seiner ersten Version a_1 anlegen und lädt diese. Er bekommt eine Kopie als Version a_3 für die Bearbeitung.

Zeitpunkt 7: Er erstellt die Variante $a_3^{'}$ und speichert diese im Repository. Die Verzweigung von Version a_1 zu den Versionen $a_2^{'}$ und $a_3^{'}$ wird im Versionsgraphen über entsprechende Kanten $(a_1, a_2^{'})$ und $(a_1, a_3^{'})$ dauerhaft und nachvollziehbar vermerkt.





Zeitpunkt 8: Der Projektleiter möchte einen verlässlichen Projektstand für die weitere Planung. Dafür will er eine Freigabe machen. Dazu muss er sich entscheiden, welche der beiden Varianten des Objekts a darin enthalten sein soll. Er entscheidet sich dafür, beide zusammenzuführen und erstellt Version a_4. Auch die Zusammenführung ist im Repository im Versionsgraphen gespeichert.

Zeitpunkt 9: Der Freigabestand soll die aktuelle Version b''_2 des Objekts b enthalten. Diese ist jedoch veraltet, da sie an eine Version a'_2 gebunden ist, von der es schon Nachfolger gibt. Bearbeiter B soll eine Aktualisierung vornehmen. Die entstandene Version b_3 ist dann an die aktuelle Version a_4 gebunden.

Zeitpunkt 10: Bei der Bildung des Freigabestandes sind Bedingungen einzuhalten: Es darf maximal nur eine Version eines Objekts enthalten sein. Außerdem müssen alle bindenden Objektversionen zum Freigabestand gehören, da sonst nicht der Zustand aller freigegebenen Objektversionen ermittelt werden kann.

Schlussfolgerungen: Die Versionierung von Objekten und Einführung von Bindungen ermöglicht das wechselseitige Arbeiten durch den Abgleich von lokalen Versionen mit Versionen des Repository, unterstützt Variantenuntersuchungen inklusive Zusammenführung, die Ermittlung von durch Änderungen betroffenen Objektversionen und die Bildung flexibler Teilmengen. Die Erhöhung der Granularität vom Dokument zum Objekt verbessert die Konsistenz bei der verteilten Bearbeitung.

Beim Vergleich zweier Zustände eines Objekts können die Werte seiner Attribute miteinander verglichen werden. Diese Unterschiede können von der Anwendung oder vom Nutzer zusammengeführt werden. Sie beinhalten jedoch eine höhere Aussagekraft als die Zeilen eines Textdokuments.

Mögliche Nachteile durch eine höhere Datenmenge bei Versionierungsansätzen lassen sich durch die Verbesserung der Hardware ausgleichen. Diese liegen nach [Beucke u. Beer 2005] vor. Jedoch existieren bisher nur theoretische Arbeiten dazu[8] und noch keine realen Systeme, die die Anwendbarkeit dieses Ansatzes im Bauplanungsprozess überprüfen.

[8] [Firmenich 2001]

1.2.7 Zusammenfassung

Kooperation: Die vorgestellten Verfahren unterstützen die *Kooperation* in unterschiedlicher Ausprägung. Dokumentenbasierte Ansätze ermöglichen nur die sequentielle oder parallele Bearbeitung. Datenbankbasierte Ansätze erlauben durch kurze Transaktionen eine quasi wechselseitige Bearbeitung. Die (Objekt-)Versionierung unterstützt alle Arten der Kooperation.

Ansatz	**Kooperation**		
	sequentiell	**parallel**	**wechselseitig**
Der letzte gewinnt	•	○	–
Sperren	•	•	–
Kurze Transaktionen	•	•	○
Versionshistorie	•	•	–
Versionierung	•	•	○
Objektversionierung	•	•	•

• mit Unterstützung ○ ohne Unterstützung – nicht möglich

Tabelle 1.1: Vergleich von Ansätzen für die verteilte Bearbeitung

Varianten: Die *Varianten*bildung ist ein wesentlicher Bestandteil der Lösungsfindung im Planungsprozess. Zur Unterstützung von Varianten müssen unterschiedliche Versionen angelegt werden können. Zudem ist es erforderlich, diese auch wieder zusammenführen zu können. Die Bearbeitungsgeschichte der Versionen ist zu speichern, um auch zu späteren Zeitpunkten noch Varianten von „alten“ Objektversionen abzweigen zu können, zum Beispiel zur Korrektur von Planungsentscheidungen. Diese Möglichkeiten sind nur bei Versionierungsansätzen verfügbar.

Ansatz	**Varianten**		
	Verzweigung	**Zusammenführung**	**Bearbeitungsgeschichte**
Der letzte gewinnt	–	–	–
Sperren	–	–	–
Kurze Transaktionen	–	–	–
Versionshistorie	–	–	○
Versionierung	•	•	•
Objektversionierung	•	•	•

• möglich ○ teilweise – nicht möglich/ nicht vorgesehen

Tabelle 1.2: Variantenunterstützung bei der verteilten Bearbeitung

Bindungen: Zur Sicherstellung der Konsistenz bei der verteilten Bearbeitung sind Bindungen zwischen abhängigen Objektversionen zu modellieren.

Je nach Art der Speicherung der Daten sind Bindungen möglich oder nicht: Zwischen Objekten in Dokumenten können im Allgemeinen aufgrund der Applikationsabhängigkeit keine Bindungen modelliert werden. In Datenbanken ist dies zwischen verschiedenen Spalten einer Tabelle möglich (z.B. durch Fremdschlüssel). Zwischen Objekten oder Objektversionen können ganz flexibel Bindungen modelliert werden. Es ist zu unterscheiden, ob Beziehungen zwischen Daten,

zwischen Mengen, die Daten enthalten, oder zwischen Versionen von Daten modelliert werden können.

Ansatz	**Bindungen**		
	Menge	**Daten**	**Version**
Dokument	–	–	–
Datenbank	•	•	○
Objektversionierung	•	•	•

• mit Unterstützung ○ ohne Unterstützung – i. Allg. nicht möglich

Tabelle 1.3: Bindungen

Redundanz und Konsistenz: *Redundante* Informationen sind mehrfach, teilweise auch in unterschiedlicher Form abgespeichert. Werden nicht alle gleichartig geändert, so entstehen In*konsistenzen*. Durch Bindungen zwischen gleichen Informationen werden diese Redundanzen explizit gemacht. Dadurch ist eine konsistente Aktualisierung möglich[9].

Redundanz tritt sehr stark bei Dokumenten mit vielen gleichen Informationen auf. Gleichzeitig sind Bindungen nur beschränkt modellierbar. Bei Datenbanken lässt sich durch Normalisierung der Tabellen die Redundanz stark beschränken. Zusätzlich lassen sich Aktualisierungsbeziehungen definieren. In versionierten Modellen existieren Redundanzen durch die Speicherung mehrerer Versionen, falls diese nicht als Änderungen gespeichert werden. Diese sind jedoch durch Bindungen und Versionsbeziehungen gekoppelt, so dass auch hier eine konsistente Aktualisierung möglich ist.

Ansatz	**Redundanz**	**Konsistenz**
Dokument	↗	↘
Datenbank	↘	↗
Objektversionierung	→	↗

↗ stark → mittel ↘ gering

Tabelle 1.4: Redundanz und ihre Auswirkung auf die Konsistenz

Nachvollziehbarkeit: Durch eine Objektversionierung ist die vollständige *Nachvollziehbarkeit* der Bearbeitung im Planungsprozess gegeben. Dies ist bei allen Ansätzen, bei denen Planungsergebnisse überschrieben werden, nicht möglich.

Ansatz	**Nachvollziehbarkeit**
Dokument	–
Datenbank	○
Objektversionierung	•

• ja ○ ohne Unterstützung – nicht möglich

Tabelle 1.5: Nachvollziehbarkeit bei der verteilten Bearbeitung

[9] [Hanff 2003a], [Hanff 2003b]

1.3 Ziel und Ansatz

1.3.1 Ziel

Ziel: Das *Ziel* dieser Arbeit ist die Entwicklung einer Systemarchitektur für verteilte Applikationen und Modelle im Bauplanungsprozess. Wie die eingangs aufgeführten Zitate deutlich machen, besteht hier einerseits Bedarf und andererseits liegen die technischen Möglichkeiten vor. Für die verteilte Bearbeitung im Bauplanungsprozess ist ein Konzept zu wählen, das alle drei Kooperationsarten unterstützt. Dies soll mit Hilfe der folgenden Beispiele verdeutlicht werden.

- **Sequentielle Bearbeitung:** Es ist offensichtlich, dass das Planungsmaterial nicht in einem Bearbeitungsschritt erzeugt wird. Die *sequentielle Bearbeitung* ist somit zu unterstützen. Dies ist schon durch die aktuelle Praxis des Dokumentenaustausches gegeben.
- **Parallele Bearbeitung:** Treten in einer Planungsphase relativ wenige Konflikte bzw. Berührungspunkte zwischen Gewerken auf, zum Beispiel bei der Tragwerksplanung und der Technischen Gebäudeausrüstung in der Detailplanung, so können die Akteure *parallel* arbeiten und erst zu einem späteren Zeitpunkt ihre Ergebnisse abgleichen.
- **Wechselseitige Bearbeitung:** Treten viele Berührungspunkte auf, wie zum Beispiel beim Stahlverbundbau, wo die verschiedenen Gewerke gemeinsame Knotenpunkte bearbeiten, so müssen die Ergebnisse der Bearbeitung relativ häufig abgeglichen werden[10].

Wie die Zusammenfassung von Seite 12 f. zeigt, ist für diese Anforderungen nur der Ansatz der Objektversionierung geeignet. Er soll im Rahmen dieser Arbeit weiter verfolgt werden.

1.3.2 Ansatz

Abstraktion: Der Austausch von Dokumenten im Bauplanungsprozess ist nicht geeignet für die Kooperation zwischen verschiedenen Akteuren. Es wird vorgeschlagen, die Detaillierung von Dokumenten hin zu Objekten zu erhöhen. Zur Unterstützung der wechselseitigen Kooperation, der Nachvollziehbarkeit und der Untersuchung von Varianten sind diese Objekte zu versionieren. Die technologischen Randbedingungen für diesen Schritt sind mit der hohen Speicherkapazität verfügbarer Hardware und leistungsfähiger Datenspeicher gegeben[11].

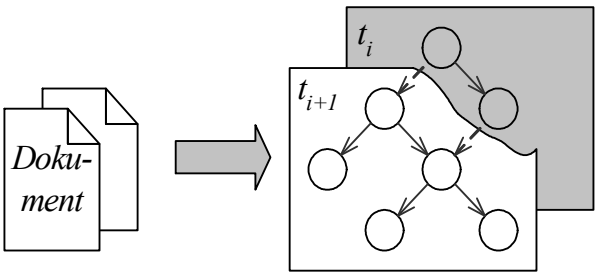


Abbildung 1.5: Ansatz – Abstraktion

[10] [Huhn 2003b]
[11] [Beucke u. Beer 2005]

Typisierung: Die Objekte unterschiedlicher Fachapplikationen sind durch verschiedene Klassen geprägt, auch wenn sie die gleiche Semantik besitzen. Eine Fachapplikation kann zumeist nicht ohne Transformation die Objekte bearbeiten, die mit einer anderen Fachapplikation erzeugt wurden. Bei Dokumenten wird der Typ durch die Dateiendung verdeutlicht. In dieser Arbeit werden Objekte zu Teilmodellen und Objektversionen zu Teilmodellversionen zusammengefasst. Jedes Teilmodell kann nur von bestimmten Fachapplikationen bearbeitet werden, dafür aber vollständig.

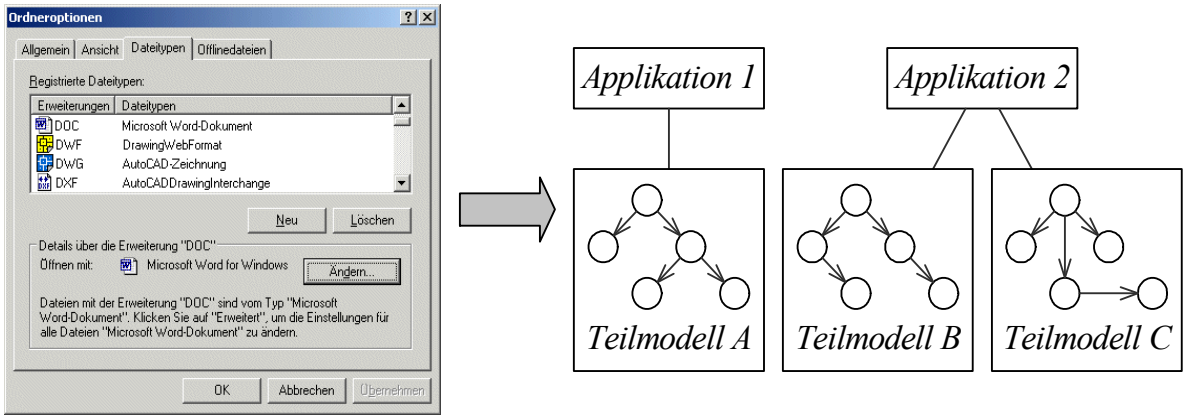


Abbildung 1.6: Ansatz – Typisierung

Modelle: Das gemeinsame Planungsmaterial wird in drei verschiedene *Modelle* eingeteilt.

- **Repository:** In einem gemeinsamen Datenspeicher, dem *Repository*, können mehrere Projekte abgelegt sein. Diese bestehen aus Teilmodellversionen, die Objektversionen und deren Eigenschaften enthalten. Sie können von mehreren Bearbeitern wechselseitig bearbeitet werden. Diese speichern dann ihre Ergebnisse als Versionen in das Repository zurück.
- **Sandbox:** In dem lokalen Datenspeicher des Bearbeiters, der *Sandbox*, ist nur ein Projekt mit von ihm ausgewählten Teilmodellen abgelegt. Die Teilmodelle enthalten Objekte und deren Eigenschaften. Es ist nicht sinnvoll, dem Bearbeiter mehr als eine Objektversion für die Bearbeitung bereitzustellen, da bestehende Fachapplikationen zumeist nicht mehrere Versionen verarbeiten können.
- **Objektmodell:** Für die Bearbeitung werden Teilmodelle der Sandbox in die Applikation geladen und liegen dort als *Objektmodell* der Applikation vor. Nach der Bearbeitung werden die Objekte der Sandbox überschrieben. Das heißt, lokale Ergebnisse werden nicht versioniert, damit die Komplexität der Modelle beherrschbar bleibt.

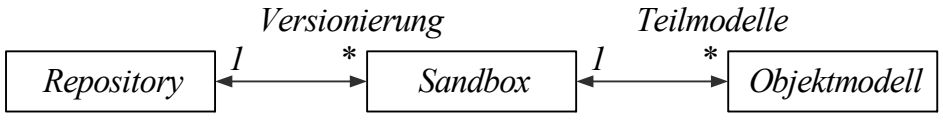


Abbildung 1.7: Ansatz – Modelle

Zwischen den Modellen bestehen Beziehungen unterschiedlicher Kardinalität.

- **Repository – Sandbox:** Ein Akteur kann an mehreren Teilmodellen mehrerer Projekte arbeiten. Für jedes Projekt ist jedoch eine eigene Sandbox anzulegen, die Teilmodelle genau eines Projekts beinhaltet. Teilmodelle können von unterschiedlichen Akteuren synchron bearbeitet werden, da sie jeweils eine eigene Version bearbeiten.
- **Sandbox – Objektmodell:** Fachapplikationen können nur Objektmodelle interpretieren, deren Klassen ihnen bekannt sind. Sie können also nicht alle Teilmodelle bearbeiten. Dafür sind mehrere Fachapplikationen nötig. Eine Fachapplikation sollte jedoch mehrere Teilmodelle gleichzeitig öffnen und unabhängig bearbeiten können.

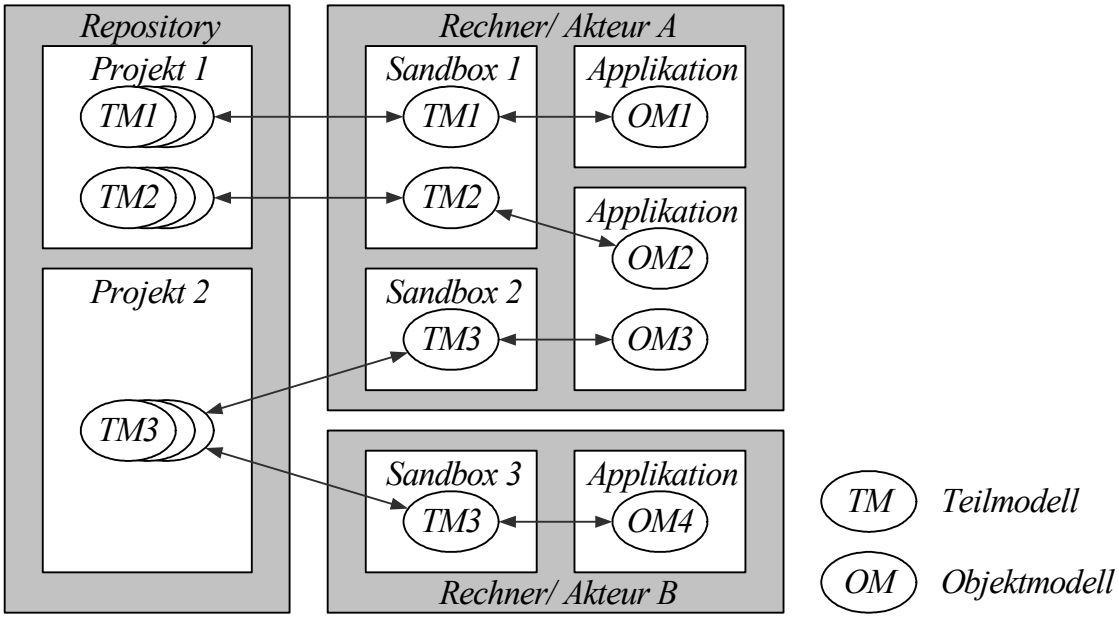


Abbildung 1.8: Ansatz – Modellbeziehungen

Teilmengenbildung: Bei dokumentenbasierten Ansätzen lassen sich *Teilmengen* von Informationen über die Aufzählung von Namen von Dokumenten in einer Verzeichnisstruktur benennen. Dies ist für den Nutzer einfach und intuitiv, da er die Verzeichnisstruktur und Dokumentbenennung selbst vorgenommen hat oder sie ihm zumindest bekannt ist. Auf der anderen Seite kann er keine Teile von Dokumenten auswählen oder bearbeiten.

Beim Ansatz der Objektversionierung ist die Zuordnung von Objektversionen zu Teilmengen nicht fest vorgegeben und kann flexibel erfolgen. Jedoch ist die Auswahl durch Aufzählung von Namen aufgrund der Komplexität des Modells bzw. der Anzahl der Objekte und deren, dem Nutzer wenig aussagekräftigen Namen hier nicht mehr sinnvoll. Der Anwender beschreibt die Menge durch die Eigenschaften ihrer zugeordneten Elemente. Deshalb wird für die Teilmengenbildung eine Auswahl der Objekte über ihre Eigenschaften vorgeschlagen. Die Eigenschaften, die der Entwickler der Klassen den Objekten gegeben hat (Attribute), sind dafür nicht geeignet, da sie dem Anwender nicht bekannt, sehr komplex und wenig aussagekräftig sind. Es wird vorgeschlagen, Objekte um äußere Eigenschaften zu erweitern. Diese werden Features genannt.

Da bestehende Applikationen nicht verändert werden sollen, können ihre Klassen im Allgemeinen nicht um Features erweitert werden. Es wird deshalb vorgeschlagen, zwischen Objekten und

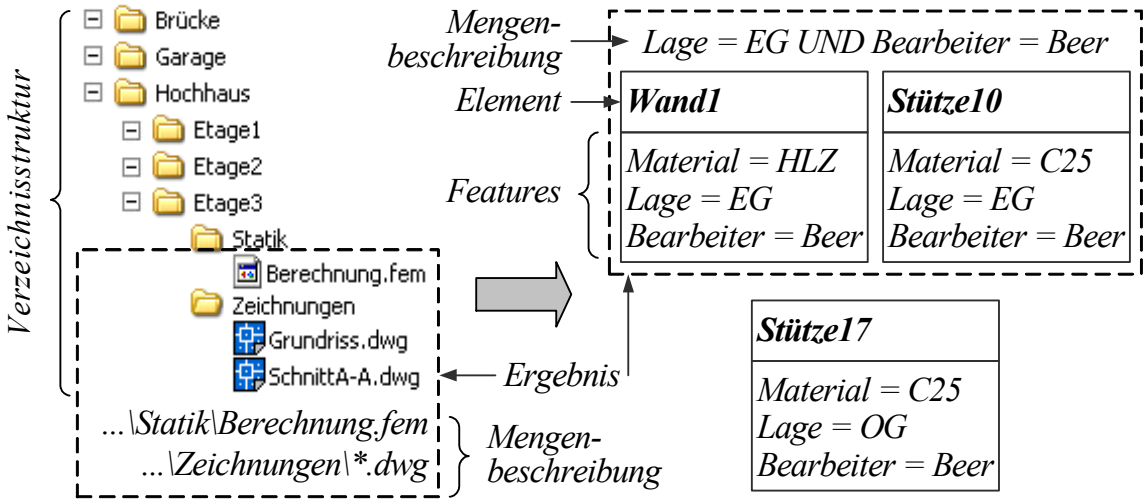


Abbildung 1.9: Ansatz – Teilmengenbildung

deren Attributen sowie Elementen und deren Features zu unterscheiden. Objekte repräsentieren den applikationsabhängigen Zustand des Datenmodells, der durch Elemente applikationsneutral hinsichtlich eines gemeinsamen Planungsmaterials erweitert wird. Der Zusammenhang zwischen Objekten und Elementen wird über einen gemeinsamen eindeutigen Identifikator hergestellt.

Beziehungen können zwischen Objekten oder Elementen bzw. deren Versionen bestehen. Beziehungen zwischen Objekten heißen Referenzen, zwischen Elementen werden sie Bindung genannt.

Referenzen sind applikationsabhängig und somit nur zwischen Objekten eines Teilmodells zulässig.

Bindungen sind applikationsneutral und können zwischen Elementen beliebiger Teilmodelle existieren. Sie werden nicht zwischen Elementen, sondern zwischen Elementversionen gebildet. Dadurch lässt sich die Dynamik des Planungsprozesses besser abbilden, da sich Bindungen zwischen Elementversionen mit deren Weiterentwicklung ändern, also hinzukommen oder entfallen können.

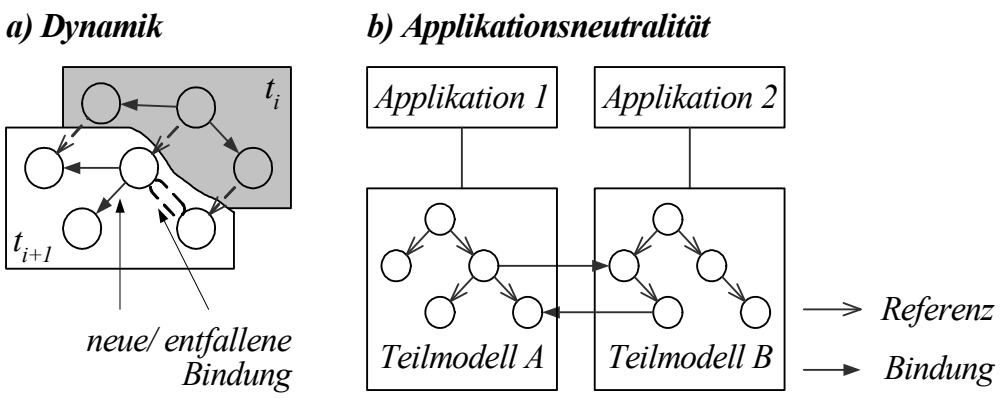


Abbildung 1.10: Ansatz – Bindungen

Redundanz: Die *Redundanz* infolge unterschiedlicher Strukturierung der Objekte durch die sie erstellenden Ingenieuranwendungen bleibt bestehen, solange kein einheitliches Objektmodell verwendet wird[12]. Über Bindungen lässt sich jedoch diese Redundanz abbilden und konsistent handhaben[13].

Konsistenz: Die *Konsistenz* der Modelle betrifft deren Inhalt und Struktur.

Der Inhalt wird durch die Anwendung von Bindungen konsistent gehalten: Bei Änderungen können betroffene Versionen mit Hilfe der Bindungen ermittelt werden.

Die Struktur der Modelle wird durch die Einhaltung von Konsistenzbedingungen erreicht. Diese werden von Operationen, die auf den Modellen definiert sind, befolgt.

Operationen bearbeiten Modelle unter Sicherung der Konsistenz oder synchronisieren Inhalte von unterschiedlichen Modellen. Bei der Bearbeitung werden Lese- und Schreibaktionen nur auf einem Modell durchgeführt, bei der Synchronisation auf mehreren Modellen.

- **Bearbeitung:** Das Objektmodell wird durch Operationen der Fachapplikation verändert. Die Sandbox lässt sich nicht direkt *bearbeiten*. Im Repository können Versionen zusammengeführt oder zu einem Freigabestand zusammengefasst werden.
- **Synchronisation:** Das Objektmodell wird mit der Sandbox durch die Serialisierung (*Speichern*) bzw. Deserialisierung (*Laden*) abgeglichen. Beim Speichern wird die Sandbox, beim Laden das Objektmodell verändert bzw. erzeugt.

  Die Sandboxen werden mit dem Repository über drei Operationen *synchronisiert*: Beim *Holen* werden Versionen in die Sandbox kopiert, beim *Übertragen* neue Versionen im Repository angelegt und beim *Aktualisieren* Versionen der Sandbox mit dem Repository aktualisiert.

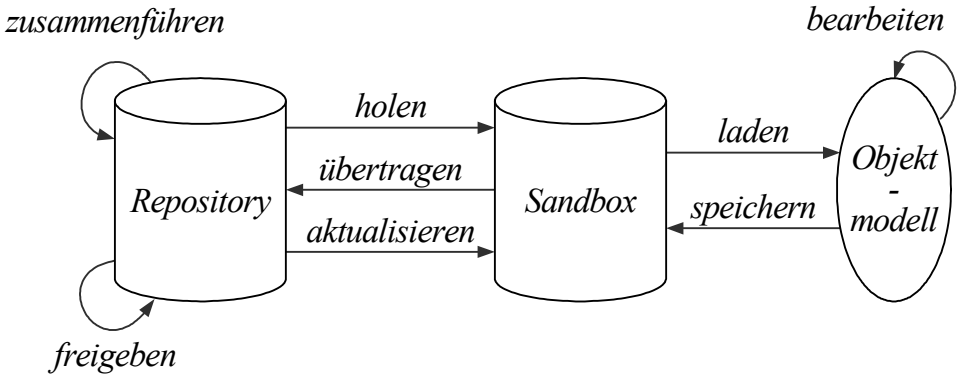


Abbildung 1.11: Ansatz – Operationen

Planungsprozess: Im *Planungsprozess* werden die auf Modelle anwendbaren Operationen in einzelnen Aktivitäten zusammengefasst. Der von [Firmenich 2001][14] abstrahierte Planungsprozess wird auf Applikationen erweitert und beinhaltet die in Abbildung 1.11 dargestellten Operationen.

[12] Zum Beispiel die IFC (industry foundation classes, *dt. „Industrie-Basisklassen“*), [IAI 2004].
[13] [Hanff 2003a]
[14] Siehe Abbildung 1.2 auf Seite 2.

- **Phase 1:** In Phase 1 holt sich der Akteur nach Auswahl einer konsistenten Teilmenge aus dem Repository diese für die lokale Bearbeitung in seine lokale Sandbox. Dort werden die lokalen Bearbeitungsergebnisse gespeichert.

- **Phase 2:** In Phase 2 bearbeitet jeder Akteur die Daten seiner lokalen Sandbox mit seinen Fachapplikationen.

  Bei synchroner Bearbeitung sich überschneidender Teilmengen ist ein Abgleich zwischen Sandboxen über das Repository möglich, indem lokale Daten mit Versionen des Repository aktualisiert werden können.

- **Phase 3:** Die Daten der lokalen Sandbox können in das gemeinsame Repository übertragen werden. Dort werden dann neue Versionen angelegt und die Bearbeitungsgeschichte der Objekte gespeichert.

  Durch den Versionierungsansatz können Varianten untersucht und diese auch zusammengeführt werden. Versionen verschiedener Objekte können zu Freigabeständen zusammengefasst und als Grundlage für weitere Planungen verwendet werden.

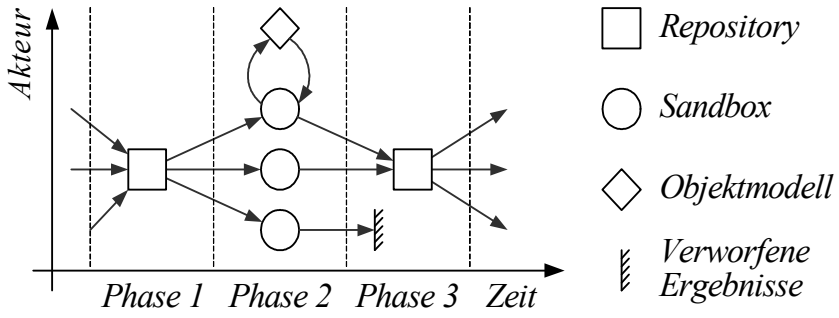


Abbildung 1.12: Ansatz – Planungsprozess

Nachvollziehbarkeit: Die Entwicklung der Versionen im Repository wird *nachvollziehbar* gespeichert. Einmal persistent gespeicherte Versionen werden nicht mehr überschrieben. Zur Unterstützung der Nachvollziehbarkeit ist jedoch auch eine Vergleichbarkeit unterschiedlicher Versionen erforderlich.

Dies ist gegeben, da die Versionen die gleiche Struktur aufweisen. Es wird ein Mechanismus vorgeschlagen, der unabhängig von der Strukturierung der Objekte Unterschiede zwischen zwei Objektversionen ermittelt. Für spezielle Anwendungen sind so Werkzeuge möglich, die die Struktur der Objekte kennen und so einen fachspezifischen Vergleich ermöglichen.

Varianten: Es wird die gesamte Entwicklung der Objektversionen, inklusive Verzweigungen, Zusammenführungen und dem Löschen von Objektversionen gespeichert. Durch Verzweigungen sind *Varianten*, aber auch nachträgliche Korrekturen möglich. Der Vergleich zweier Zustände unterstützt die fachliche Entscheidung über weiter zu verfolgende Lösungen zum Beispiel bei der Zusammenführung.

1.4 Vorgehensweise

Vorgehensweise: Die Arbeit gliedert sich in die Kapitel *Systementwurf*, *Stand der Technik und Forschung*, *Mathematische Beschreibung*, *Umsetzungskonzept* und *Pilotimplementierung*.

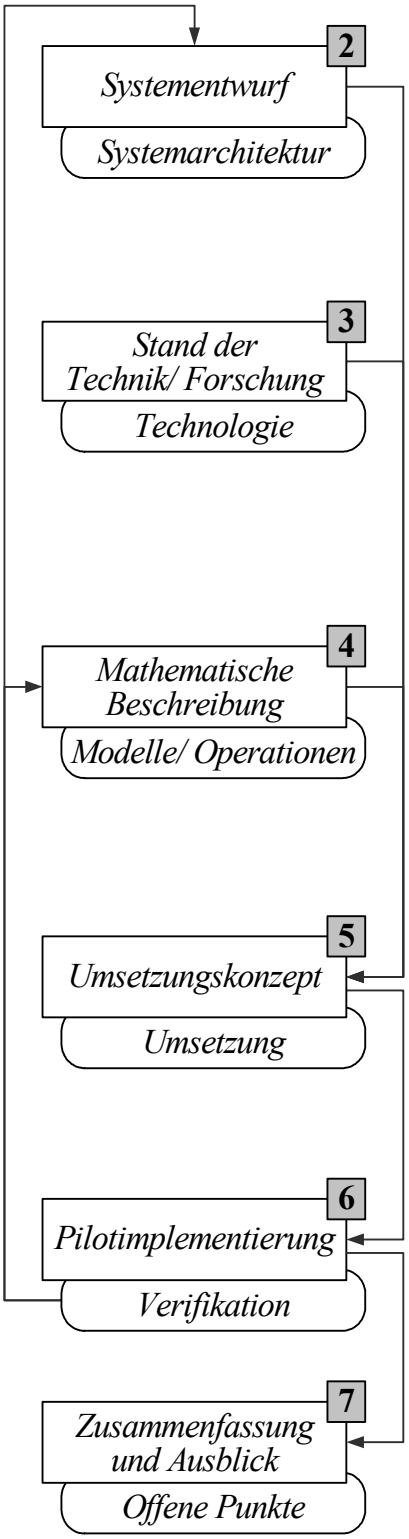


Abbildung 1.13: Vorgehen

Systementwurf: Beim *Systementwurf* wird eine Systemarchitektur für verteilte Applikationen und Modelle im Bauplanungsprozess konzipiert. Diese ist unabhängig von vorhandener Technologie.

Stand der Technik und Forschung: Beim *Stand der Technik und Forschung* werden geeignete Technologien und Konzepte für die Umsetzung vorgestellt. Das betrifft die im Systementwurf identifizierten Elemente der Systemarchitektur.

Mathematische Beschreibung: Die in der Systemarchitektur definierten Modelle und Operationen werden unabhängig von der Umsetzung *formal beschrieben*. Dies dient zum einem dem klaren Verständnis und zum anderen einer dauerhaften Beschreibung des Konzepts.

Umsetzungskonzept: Für die Umsetzung der Modelle und Operationen des Systementwurfes mit Hilfe geeigneter Technologie wird ein *Umsetzungskonzept* vorgestellt.

Pilotimplementierung: Bei der Vorstellung einer Pilotanwendung werden konkrete Software-Produkte ausgewählt. Die *Pilotimplementierung* dient der Verifikation des Systementwurfes und der mathematischen Beschreibung.

Zusammenfassung und Ausblick: Die Arbeit schließt mit einer *Zusammenfassung* und einem *Ausblick* auf offene Punkte.

Kapitel 7

Zusammenfassung und Ausblick

7.1 Zusammenfassung

Problemstellung: Der Planungsprozess im Konstruktiven Ingenieurbau ist gekennzeichnet durch drei sich zyklisch wiederholende Phasen: die Phase der Aufgabenverteilung, die Phase der parallelen Bearbeitung mit entsprechenden Abstimmungen und die Phase der Zusammenführung der Ergebnisse. Verfügbare Planungssoftware unterstützt überwiegend nur die Bearbeitung in der zweiten Phase und den Austausch der Datenbestände. Diese werden in Form von Dokumenten modelliert. Zwischen den Dokumenten der verschiedenen Applikationen können im Allgemeinen keine Beziehungen modelliert werden, so dass bei Änderungen die Verfolgung der Auswirkungen aufwändig ist und das Planungsmaterial oft inkonsistent wird. Mit dem dokumentenbasierten Ansatz ist kein wechselseitiges Bearbeiten des gleichen Planungsmaterials möglich. Auch für die einfachste Form der Kooperation - dem asynchronen Arbeiten - ergeben sich Nachteile, wenn Dokumente in jedem Arbeitsschritt überschrieben werden, da die Nachvollziehbarkeit und der Bezug auf verlässliche Quellen nicht mehr möglich ist.

Ziel und Ansatz: *Ziel* dieser Arbeit ist die Entwicklung einer Systemarchitektur, die in ihrem Grundsatz alle Phasen der verteilten Bearbeitung und Arten der Kooperation berücksichtigt.

Als *Ansatz* wird das gemeinsame Arbeitsmaterial der Beteiligten nicht als Dokumentmenge, sondern als Menge von Elementversionen und Objektversionen und deren Beziehungen abstrahiert. Die Trennung von Objekten und Elementen resultiert aus der applikationsspezifischen Implementierung der Objekteigenschaften (Attribute), über die keine gemeinsame Struktur des Planungsmaterials gebildet werden kann. Mit den Elementeigenschaften (Features) können applikationsübergreifend Bindungen, Versionsbeziehungen sowie spezifische Applikations- und Anwendereigenschaften modelliert und so das Objektmodell ohne Änderung der Klassen erweitert werden.

Für die Bearbeitung einer Aufgabe werden auf Basis der Features Teilmengen gebildet, für deren Elemente neue Versionen abgeleitet und in einen privaten Arbeitsbereich geladen werden. Die Teilmengenbildung beruht auf einer Mengenalgebra und erweitert diese um Operationen in Verbindung mit den Eigenschaften der Elemente, den Features. In dieser Arbeit werden weitere Operationen in Verbindung mit der Versionierung und den Bindungen beschrieben sowie ein Konzept für fachspezifische Erweiterungen gezeigt. Die Teilmengenbildung erfolgt mit Hilfe einer Sprache, für die bei vorliegender Grammatik ein Interpreter automatisch erzeugt

werden kann. Mit diesem können Ausdrücke der Sprache interpretiert und in eine Sprache übersetzt werden, die auf dem Datenspeicher, der dem persistenten Datenmodell zugrunde liegt, angewendet werden kann. Durch Verwendung von Standards, zum Beispiel SQL oder JDOQL, kann vom konkreten Datenspeicher oder sogar dem Typ des Datenspeichers abstrahiert werden.

Die Bearbeitung wird auf Operationen zurückgeführt, mit denen das gemeinsame Arbeitsmaterial konsistent gehalten wird. Diese Konsistenz wird auf struktureller Ebene mit Hilfe der Mengen- und Relationenalgebra formuliert und bei der Implementierung der Operationen berücksichtigt. Die Operationen lehnen sich an Versionsverwaltungssysteme aus dem Bereich der Softwareentwicklung an. Sie sind auf diesem Gebiet Stand der Technik und unterstützen die verteilte Entwicklung von Software auf Basis der Versionierung des Quellcodes in einer Historie.

Systementwurf: Der *Systementwurf* beschreibt allgemein den Aufbau eines verteilten Softwaresystems für die verteilte Bearbeitung im Bauplanungsprozess. Er besteht aus Komponenten mit Schnittstellen, über die Daten ausgetauscht werden und Operationen, die den Komponenten zugeordnet sind und deren konsistente Bearbeitung sichern. Operationen bearbeiten Modelle und synchronisieren sie untereinander.

Es wird ein Ebenenkonzept vorgeschlagen, das die Komponenten vertikal in die Aspekte lokale und zentrale Anwendung sowie horizontal in Anwendung und Modell einteilt. Die lokale Anwendung lässt sich in eine fachliche und eine Verwaltungskomponente, das Modell in eine logische und eine physische Ebene teilen.

Die lokale Fachanwendung ist eine bestehende Ingenieuranwendung, die für die verteilte Bearbeitung erweitert wird. Die lokale Verwaltung übernimmt der Workspace, der die Verteilung durch Kommunikation mit der zentralen Anwendung – dem Project – realisiert. Auf Modellebene werden die Daten der Fachapplikation im Objektmodell, die Elemente des Workspace in der Sandbox und die Objekt- und Elementversionen des Project im Repository logisch strukturiert. Das zugrunde liegende physische Modell ist der Arbeitsspeicher bzw. ein Datenspeicher.

Stand der Technik und Forschung: Der Systementwurf ist unabhängig von Technologie beschrieben, um die Dauerhaftigkeit des Konzeptes zu gewährleisten. Beim *Stand der Technik und Forschung* werden bestehende Technologien, Standards und Konzepte für die Komponenten und Operationen des Systementwurfes beschrieben, die grundsätzlich für die Umsetzung geeignet sind.

Mathematische Beschreibung: Der Aufbau der Modelle und die Semantik der Operationen wird unabhängig von Technologie formal mit Hilfe der Mengen- und Relationenalgebra *beschrieben*. Es wird auch vom Systementwurf abstrahiert, so dass dieser modifiziert werden kann. Basis ist jedoch der Ansatz der Versionierung und der Trennung von Objekten und Elementen.

Es zeigt sich, dass die applikationsspezifischen Beziehungen zwischen Objekten (Referenzen) nicht zusätzlich zu den Objekteigenschaften als Features zu modellieren sind. Dies liegt vor allem an der Größe und Komplexität von Ingenieurmodellen. Als Lösung werden Teilmodelle vorgeschlagen. Zwischen Objekten verschiedener Teilmodelle dürfen keine Referenzen bestehen. Dies ist vergleichbar mit Dokumenten, jedoch viel flexibler: Bindungen zwischen Elementen verschiedener Teilmodelle sind möglich, so dass applikationsübergreifend Beziehungen modelliert werden können, die auch bei der Selektion berücksichtigt werden. Selektionsmengen sind eine Vereinigung von Teilmodellversionen, da garantiert werden muss, dass alle referenzierten Objekte durch die Selektionsmenge erfasst wurden.

Umsetzungskonzept: Für das *Umsetzungskonzept* wird eine Kombination von Eigenentwicklung und Nutzung von Standardtechnologie vorgeschlagen. Für die Versionierung der Objekte kommt ein Versionsverwaltungswerkzeug zum Einsatz, das die Änderungen an den Objektversionen sehr effizient über das Netzwerk überträgt und speichert. Da solche Systeme jedoch keinen geeigneten und performanten Mechanismus zur Teilmengenbildung bieten und keinen allgemeinen Versionsgraphen verwalten, werden parallel dazu die Elementversionen in einer relationalen Datenbank gespeichert. Eine relationale Datenbank wurde gewählt, da sie für die Verwaltung von Massendaten mit einfacher Struktur sehr gut geeignet ist und eine standardisierte Zugriffssprache (Schnittstelle) besitzt. Beides liegt vor: Durch die Versionierung entstehen viele Daten und das Element-Feature-Modell ist sehr einfach und generisch über wenige Mengen und Relationen beschreibbar.

Für die Auswahl eines Datenspeichers spielt die Komplexität eine wesentliche Rolle. Diese beschreibt den Zusammenhang zwischen Eingangsdaten und benötigten Ressourcen eines Algorithmus. Eingangsdaten sind im Allgemeinen die Modellgröße, die wichtigsten Ressourcen sind die erforderliche Zeit und der benötigte Arbeitsspeicher. Die Modellgröße wird an Beispielen für die Sandbox und das Repository abgeschätzt, so dass geeignete Datenspeicher ausgewählt werden können.

Pilotimplementierung: Die Systemarchitektur wird *pilothaft* mit ausgewählter Technologie *implementiert*. Dies erfolgt in der Umgebung des Internet in der Sprache Java. Als Fachapplikation wird die CAD-Applikation **CAD**EMIA verwendet, da sie ebenfalls in der Sprache Java implementiert und sehr leicht zu erweitern ist. Die Objektversionierung wird von dem aktuellen Versionsverwaltungssystem Subversion übernommen, die Elementversionierung erfolgt mit der Feature-Logic in einer Oracle-Datenbank. Die Feature-Logic bietet Möglichkeiten zur Synchronisierung des Feature-Logic-Modells mit dem physischen Datenspeicher. Eine Object Version Query Language (OVQL) genannte eigene Sprache erweitert die Feature-Logic für die Selektion.

Für die Operationen werden exemplarisch zwei Szenarien gezeigt, die die Vorteile des Ansatzes darlegen.

Erkenntnisse: In der Arbeit wird der Wandel vom dokumentenbasierten zum versionsbasierten Ansatz mit Objekten und Elementen vorgeschlagen. Es zeigen sich Verbesserungen hinsichtlich der Modellkonsistenz, der Modellbearbeitung und der Prozessunterstützung.

Einmal gespeicherte Elementversionen werden vereinbarungsgemäß nicht aus dem gemeinsamen Modell ausgetragen und ändern sich nicht mehr. Damit bleiben Beziehungen zwischen Elementversionen gültig und eignen sich zur Referenzierung durch andere Elementversionen. Änderungen beziehen sich nur auf abgeleitete Elementversionen. Dadurch wird die Komplexität des versionierten Modells beherrschbar, da eine umfassende und sofortige Aktualisierung nicht erforderlich ist.

Technische Lösungen werden iterativ – in der Regel über einen längeren Zeitraum – erarbeitet. Die Zwischenstände müssen nicht konsistent sein, da die Bearbeitung unter Ausschluss der Öffentlichkeit in einem geschützten Arbeitsbereich erfolgt. Der Ingenieur entscheidet, ob das Ergebnis seiner Bearbeitung im gemeinsamen Modell gespeichert werden soll. Durch den Abgleich der verschiedenen Versionen der einzelnen Planer über das gemeinsame Repository wird

die synchrone Kooperation ermöglicht. Dies ist in der Praxis notwendig und muss daher von einem modernen verteilten System unterstützt werden.

Beim Prozess der Bearbeitung des Planungsmaterials treten Änderungen auf. Diese sind durch die Versionierung nachvollziehbar und können später analysiert werden. Im Bauprozess vorgeschriebene Revisions- und Freigabestände sind abbildbar. Die technische Lösungfindung wird durch Verzweigung des Versionsgraphen und damit die Untersuchung alternativer Varianten unterstützt. Nach Änderungen müssen die davon betroffenen Planer benachrichtigt werden. Bei einem versionierten Modell können die zwei Arten der Benachrichtigung nach dem Prinzip der Hol- und Bringschuld umgesetzt werden.

Bewertung: Der Ansatz der Abstrahierung des Planungsmaterials als Objekt- und Elementversionen unterstützt die Arbeitsweise im Bauplanungsprozess und ist damit für eine praktische Anwendung geeignet. Die grundlegenden Konzepte konnten mit dieser Arbeit beschrieben werden. Die entwickelte Systemarchitektur ist für eine professionelle Umsetzung geeignet. Dafür verwendbare Technologien wurden beschrieben. Die genaue Semantik der Modelle und Operationen wurde mit einer formalen Beschreibung definiert. Das Umsetzungkonzept bringt Systemachitektur, formale Beschreibung und Technologie zusammen. Hier können in Zukunft jedoch Änderungen erfolgen, die sich aus fortgeschrittenen Technologien oder anderen Umsetzungsstrategien ergeben. Aus diesem Grund muss die auf dem Umsetzungskonzept basierende Pilotanwendung als reine Testumgebung betrachtet werden. Die Einbeziehung des Nutzers steht als eine wesentliche Aufgabe noch zwischen dieser Pilotimplementierung und einem praktischen Prototypen.

7.2 Ausblick

Ausblick: Wichtige Punkte, die in dieser Arbeit offen geblieben sind oder nicht vollständig behandelt werden konnten, betreffen die Beherrschung der Komplexität, die Sicherung der Konsistenz, die Integration unterschiedlicher Fachanwendungen sowie die Verknüpfung von Produkt- und Prozessmodell.

Beherrschung der Komplexität:[1] Für die Selektion benötigt der Nutzer Kenntnis über die Struktur und den Inhalt des versionierten Modells in einer für ihn verständlichen Informationstiefe. Er muss wissen, welche neuen Versionen von anderen Planern erstellt wurden, welche eigenen Versionen er aktualisieren muss, wie er Varianten ermitteln und zusammenführen kann. Für diese Zwecke sind geeignete und neue Formen der Nutzerinteraktion zu erforschen. Dies betrifft auch die Darstellung des gesamten oder von Teilen des versionierten Modells mit Hilfe grafischer Nutzeroberflächen. Dabei sind in verschiedenen Kontexten unterschiedliche Darstellungsarten und -tiefen zu wählen: Objekte und Elemente, deren Versionen mit Versionsbeziehungen, Teilmodelle und deren Versionen sowie Bindungen sind geeignet darzustellen.

Dies betrifft auch die Entwicklung von für Domänen des Konstruktiven Ingenieurbaus geeigneten Fachsprachen, die auf der Feature-Logic und der OVQL basieren und fachspezifische Semantik in Operationen kapseln. Diese Sprache sollte für ungeübte Anwender mit Elementen einer grafischen Nutzeroberfläche verknüpft werden, aber auch über eine Kommandosprache Anwendern und Makro-Interpretern zur Verfügung stehen.

[1] [Richter 2006]

Sicherung der Konsistenz: Durch die Trennung der Versionierung von Objekten und Elementen mit einem Versionsverwaltungssystem bzw. der Feature-Logic in einer relationalen Datenbank sind die beiden Modelle zu synchronisieren. Dies betrifft die Operationen, die zwischen Sandbox und Repository Daten austauschen. In der Arbeit wurden dafür Transaktionen vorgeschlagen, die sich gegenseitig beeinflussen: Dieses Transaktionskonzept ist zu vertiefen und in einer Pilotanwendung zu überprüfen.

Integration verschiedener Fachanwendungen: Das Konzept zur verteilten Bearbeitung ist anwendungsübergreifend. Im Rahmen der Arbeit wurde jedoch nur eine Fachapplikation aus dem Bereich CAD *integriert*. Die Integration ist für weitere Fachanwendungen vorzunehmen, um auch die Definition von Bindungen zwischen Modellen unterschiedlicher Anwendungen zu untersuchen. Für das anwendungsübergreifende Arbeiten ist jedoch eine Standardisierung der Objektmodelle der Anwendungen erforderlich. Dies kann auf Basis der Objektmodelle[2] oder der sie erzeugenden Operationen[3] erfolgen. Entsprechende Forschungsarbeiten liegen vor[2] oder sind in Bearbeitung[3].

Verknüpfung von Produkt- und Prozessmodell: Es bestehen Beziehungen zwischen *Produkt- und Prozessmodell*: Die Aktionen des Prozessmodells bearbeiten das Produktmodell, dessen Daten wiederum den Ablauf des Prozessmodells beeinflussen können. Der Zusammenhang zwischen versioniertem Produktmodell und Prozessmodell wurde in der Literatur bisher noch nicht beschrieben. Ansätze einer flexiblen Verknüpfung zu unversionierten Produktmodellen[4] können jedoch erweitert werden.

[2] [Weise u. a. 2004], [Nour 2006]
[3] [Firmenich 2004b], [Koch 2005]
[4] [Heinrich 2005], [Heinrich 2006] beschreibt eine Verbindung über eine Kennzeichnung von Objekten im Produktmodell und deren Verwendung im Prozess über Mengen von Objekten, die durch Kennzeichnungsanfragen beschrieben werden. Dies kann mit Features und Selektionstermen umgesetzt werden.

Anhang A

Verzeichnisse

A.1 Symbolverzeichnis

Schriftarten und -formen

Ansatz	Hervorhebung (kursiv)
`Class`	Klasse (Quellcode, kursiv)
`Code`	Quellcode (nichtproportional)
„*Zitat*“	Zitat (kursiv, Hochkommata)

Daten

	Beziehungen
	Daten
	Dokument
	Kardinalität (z.B. beliebig viele)
	Netzwerk
	Ordner, Verzeichnis
	Tabelle

Objekte und Elemente

	Atomarer Attributwert
	Attribut, Referenz
	Element (Kleinbuchstabe)
	Elementversion (Versionsindex)
	Feature, Bindung
	Featurewert
	Objekt (Kleinbuchstabe)
	Objektversion (Versionsindex)
	virtuelle Elementversion

Mengen

A	Attributnamenmenge
α	Attributwertemenge
M	Elementmenge
F	Featurenamenmenge
β	Featurewertemenge
$\langle a_n \rangle$	(Geordnete) Folge von Elementen a_i mit $1 \leq i \leq n$ $\langle a_n \rangle = (a_1, \ldots, a_i, \ldots, a_n)$
L_i	Freigabe
L	Freigabemenge
$\emptyset$	Leere Menge
A, B	Menge (Großbuchstaben)
Δ	Menge virtueller Elementversionen
(a_n)	Menge von Elementen a_i mit $1 \leq i \leq n$, $(a_n) = \{a_1, \ldots, a_i, \ldots, a_n\}$
$\mathbb{N}$, $\mathbb{R}$, $\mathbb{S}$	Menge der natürlichen, rationalen Zahlen, Zeichenfolgen
O	Objektmenge
T_i	Teilmodell
T	Teilmodellmenge

Index

$_1$, $_i$	Versionsindex, Element einer Menge (tiefgestellte Zahl)
$_R$	Repository
$_S$	Sandbox

Relationen

$a : M \rightarrow N$	Abbildung a aus der Menge M auf die Menge N
$<\boldsymbol{B}>_t$, $\longrightarrow\!\triangleright$	Abhängigkeiten
$\boldsymbol{A}$	Attributrelation
$a : \boldsymbol{A} \rightarrow A$	Attributnamenabbildung
$\boldsymbol{B}$, $\longrightarrow\!\!\blacktriangleright$ . .	Bindungen
m, $\mathrel{-}\!\cdot\!\blacktriangleright$ . .	Elementversionsabbildung
$\varepsilon : \mu \rightarrow \Omega$.	Elementversionszuordnung
$e : M \rightarrow O$	Elementzuordnung
$f : \boldsymbol{F} \rightarrow F$	Featurenamenabbildung
$\boldsymbol{F}$	Features
$<\boldsymbol{G}>_t$, $\longrightarrow\!\!\!\!\triangleright\!\triangleright$	gefrorene Abhängigkeiten
$\boldsymbol{G}$, $\longrightarrow\!\!\blacktriangleright\!\blacktriangleright$ . .	gefrorene Bindungen
ω, $\mathrel{-}\!\cdot\!\blacktriangleright$	Objektversionsabbildung
$\boldsymbol{R}$	Referenzen
$\boldsymbol{R}$, $\boldsymbol{S}$	Relation (Großbuchstaben, fett)
τ, $\mathrel{-}\!\cdot\!\blacktriangleright$	Teilmodellversionsabbildung
$<\boldsymbol{R}>_t$	Transitive Hülle der Relation $\boldsymbol{R}$
$\boldsymbol{V}$, $\mathrel{-}\!\!\blacktriangleright$ . .	Versionsrelation
$<\boldsymbol{V}>_t$, $\mathrel{-}\!\!\!\triangleright$	Versionsrelation (Hülle)

Graphen

$\mathfrak{G}$	Graph (gotischer Großbuchstabe), $\mathfrak{G} = (V, \boldsymbol{E}, n)$, Knotenmenge V, Kantenmenge $\boldsymbol{E}$, Kantennamen n

Operatoren

$\underset{x}{\forall}\, Y$	All-Quantor: für alle Elemente x ist die Bedingung Y erfüllt
$X \Leftrightarrow Y$ . . .	Äquivalenz
$x \in M$	x ist ein Element der Menge M
$\underset{x}{\exists}\, Y$	Exist-Quantor: es existiert ein x, das Y erfüllt
$X \Rightarrow Y$ . . .	Implikation
$M \subseteq N$ . . .	M ist Teilmenge von N
$\boldsymbol{R} \sqsubseteq \boldsymbol{S}$	$\boldsymbol{R}$ ist Teilmenge von $\boldsymbol{S}$
$\cup, \cap, \setminus$	Vereinigung, Schnitt, Differenz (Mengen)
$\sqcup, \sqcap, \setminus$	Vereinigung, Schnitt, Differenz (Relationen)

Feature-Logic

$f\uparrow$, $f\hat{}$	Divergenz (Elemente ohne Feature f)
$f : D$	Existenz (Elemente mit Feature f)
$f(S)$, $S.f$.	Extraktion (Elemente aus S mit Feature f)
$\{\}$	leere Menge
$\sim x$	nicht x
$[x, y]$	Schnittmenge von x und y
$f : S$	Selektion (Elemente mit Wert aus S für Feature f)
$f == g$ . . .	Übereinstimmung (Elemente mit gleichen Werten für Feature f, g)
$\{x, y\}$	Vereinigungsmenge von x und y
$f! = g$	Verschiedenheit (Elemente mit unterschiedlichen Werten für Feature f, g)

Sprache

*	beliebig viele
?	kein oder ein
+	mindestens ein
ˆ	nicht
12	Nichtterminal
$a \mid b$	a oder b
12	Terminal
a, b	a und b

Programmcode, Ablauf

○	Ende
$\mathrel{-}\!\!\triangleright$	Implementierung (Schnittstelle)
`void`	keine Rückgabe (leer)
`meth()` . . .	Methode
`+`	öffentlich (Sichtbarkeit)
`-`	privat (Sichtbarkeit)
●	Start
$\longrightarrow\!\triangleright$	Vererbung

A.2 Abkürzungen

API (application programming interface, *dt. „Anwendungsprogrammierschnittstelle“*) 45

ARX (AutoCAD runtime extension, *dt. „AutoCAD-Laufzeiterweiterung“*) 45

CORBA (common object request broker architecture, *dt. „Allgemeine Architektur zur Vermittlung von Anfragen an Objekte“*) 60

CVS (concurrent versions system, *dt. „nebenläufiges Versionssystem“*) 48

DBMS (*data base management system*, *dt. „Datenbankmanagementsystem“*) 71

DTD (document type definition, *dt. „Dokumenttypdefinition“*) 75

ERM (entity relationship model, *dt. „Gegenstand-Beziehung-Modell“*) 72

GUID (globally unique identifier, *dt. „global eindeutiger Identifikator“*) 47

IDL (interface definition language, *dt. „Schnittstellendefinitionssprache“*) 60

IFC (industry foundation classes, *dt. „Industrie-Basisklassen“*) 18

JDO (Java data objects, *dt. „Java-Datenobjekte“*) 57

JDOQL (JDO query language, *dt. „JDO-Anfragesprache“*) 57

MAS (multi agent systems, *dt. „Multiagentensysteme“*) 62

MVC (model view controller, *dt. „Modell-Präsentation-Steuerung“*) 24

ODL (object definition language, *dt. „Objektdefinitionssprache“*) 74

OQL (object query language, *dt. „Objektzugriffssprache“*) 74

ORB (object request broker, *dt. „Objektanfragevermittler“*) 60

OVQL (object version query language, *dt. „Objektversionsabfragesprache“*) 122

POID (persistent object identifier, *dt. „persistenter Objektidentifikator“*) 28

PVID (persistent version identifier, *dt. „persistener Versionsidentifikator“*) 120

QL (query language, *dt. „Abfragesprache“*) 122

RCS (revision control system, *dt. „Revisionsverwaltungssystem“*) 48

RMI (remote method invocation, *dt. „entfernter Methodenaufruf“*) 60

RPC (remote procedure call, *dt. „entfernter Prozeduraufruf“*) 60

SCCS (source code control system, *dt. „Quellcode-Verwaltungssystem“*) 48

SCM (software configuration management, *dt. „Softwarekonfigurationsmanagement“*) . . 48

SetDsc (set descriptor, *dt. „Mengenbeschreiber“*) 126

SQL (structured query language, *dt. „strukturierte Anfragesprache“*) 55

TOID (transient object identifier, *dt. „transienter Objektidentifikator“*) 25

URL (uniform resource locator, *dt. „Internetadresse“*) 63

VCS (version control system, *dt. „Versionsverwaltungssystem “*) 113

XData (extended data, *dt. „erweiterte Daten“*) 64

XML (extensible markup language, *dt. „erweiterbare Auszeichnungssprache“*) 56

XPath (XML path language, *dt. „XML-Adressierungssprache“*) 56

XQuery (XML query language, *dt. „XML-Zugriffssprache“*) 75

XSD (XML schema definition, *dt. „XML-Schema-Definition“*) 75

XSLT (extensible stylesheet language transformation, *dt. „XSL-Transformation“*) 56

A.3 Tabellen

A.4 Beispiele

A.5 Quellcode

A.6 Abbildungen

A.7 Zitate

Aho u. a. 2002 58
Alternate Computing Solutions 2005 135

Barker 1992 72
Baumeister u. Petrowski 2002 64
Beer 2002 ..... 45, 64
Beer u. Firmenich 2003 ..... 91, 109
Berczuk u. Appleton 2002 48
Beucke 2000 25
Beucke 2002 24
Beucke u. a. 2002 ..... iii, 2
Beucke u. a. 2004a ..... iii, 2
Beucke u. a. 2004b iii
Beucke u. Beer 2005 ..... 1, 11, 14
Beucke u. Freundt 2000 24
Beucke u. Heinrich 2002 71
Bierman 2000 118
Bilchuk u. Pahl 2004 76
Bilchuk 2005 ..... 76, 77
Blandy u. a. 2002 48
Booch 1993 25
Booch 1995 ..... 21, 25
Booch u. a. 2005 ..... 21, 23
Bretschneider 1998 ..... 3, 4
Broy 1998 ..... 21, 187
Bucher u. Müller 2004 iii
Budszuhn 2005 ..... 48, 49, 63, 67

Carnegie Mellon Software Engineering Institute 2005 21
Cattell 2000 74
Cederqvist 2002 48
Claus u. Schwill 2003 ..... 58, 187
Collins-Sussman u. a. 2005 48

Date 2000 72
Dehnhardt 1999 72
DIN 6789-5 1995 68
DIN 6789-7 2003 68
Downing 1998 60

Eckel 1998 ..... 46, 113
Elmasri u. Navathe 2000 73
Emmerich 2003 60

Firmenich 2000 ..... 61, 125
Firmenich 2001 . 1, 2, 4, 11, 18, 38, 53, 54, 79, 81, 88, 89, 96–98, 102, 120, 143
Firmenich 2002a 22, 35, 79, 126
Firmenich 2002b 79
Firmenich 2004a 2
Firmenich 2004b 3, 155
Firmenich 2004c 3
Firmenich 2004d 2
Firmenich 2005 46
Firmenich 2006 ..... 46, 114, 135, 136
Firmenich u. a. 2004 iii
Firmenich u. a. 2005 ..... 22, 35, 68, 115, 118
Fish 2005 48
Flanagan 2003 46
Fogel u. Bar 2002 48
Foley u. a. 2002 24
Forman u. Forman 2004 47
Fowler u. Scott 1999 23
Friedl 2003 187
Frischalowski u. Pallmer 2004 48
Furix.com 2005 59

Gamma 2003 ..... 21, 189
Gumm u. Sommer 2002 ..... 21, 72, 187

Hanff 2003a 13, 18, 85, 109
Hanff 2003b 13, 109
Hartmann u. a. 2004 62
Hass 2003 48
Hauser u. Löwer 2004 60
Heinrich 2005 155
Heinrich 2006 155
Herold u. Meyer 1995 48
Heuer u. Saake 2000 ..... 72, 74
Hopcroft u. a. 2002 ..... 187, 192
Horstmann u. Cornell 2003 ..... 46, 61, 66, 113

Huhn 2003a 75, 118
Huhn 2003b 14
Hunt u. McIlroy 1976 59

IAI 2004 . . . . . 3, 18

Jordan u. Russel 2003 57

Kazakos u. a. 2002 75
Kemper u. Eickler 2004 . . . . . 72, 74
Kim 1991 74
Kitz 2004 21
Klauer 2005 2
Klettke u. Meyer 2003 75
Klingelhöller 2001 8
Koch 2005 . . . . . 3, 155
Krüger 2002 . . . . . 46, 113

Laabs 1998 25
Langner 2002 60
Lausen 2005 . . . . . 69, 75
Lin 1996 48
Loney u. Theriault 2001 135

MacKenzie u. a. 2003 59
Matthiessen u. Unterstein 2003 . . . . . 55, 72
McAuley 2000 45
Meißner u. Rüppel 1999 . . . . . iii, 1, 2
Meißner u. a. 2000 2
Meißner u. Rüppel 2003 2
Meißner u. a. 2005 62
Meyer 1990 25
Müller u. a. 2003 8
Myers 1986 59

Neufein 2003 48
Nour 2006 155
Nussbaumer u. Mistelbacher 2003 . . 56, 57, 69, 75

Object Database Management Group 2005 74
Olbrich 1998 25

Pahl u. Beucke 2000 . . . . . 4, 24, 34
Pahl u. Damrath 2000 . . . . . 32, 53, 79, 185
Purdy 2001 48

Ramunno 2005 . . . . . 59, 68
Reinhard u. Soeder 2001 . . . . . 79, 185
Richter 2003 122
Richter 2006 154
Rüppel u. Meißner 2000 2
Rüppel u. a. 2002 62
Rüppel u. Meißner 2003 2
Ruh u. a. 2001 60
Rumbaugh u. a. 1991 25

Saake u. a. 1997 74
Scheid 2000 . . . . . 79, 185
Schicker 2000 . . . . . 55, 72
Schröder 1990 48
Sleepycat 2005 51
Smolka 1992 53
Stroustrup 2002 45
Stubblebine 2004 187
Sun Microsystems, Inc. 2003a 70
Sun Microsystems, Inc. 2003b 58

Tanenbaum 2003 60
Theiß 2005 62
Theiß u. Bilek 2003 62
tmate.org 2005 135
Turau 1996 . . . . . 66, 79

Ullenboom 2003 . . . . . 46, 66, 113

Vesperman 2004 48
van der Vlist 2003 75

W3C 2005 75
Wagenknecht 2003 192
Weiß 2004a 48
Weiß 2004b 48
Weiß u. Jakob 2005 62
Weise u. a. 2004 . . . . . 126, 155
WeltWeitBau 2005 59
Willenbacher 2002 62

Zeller 1997 53

A.8 Literaturverzeichnis

[Aho u. a. 2002]
AHO, A. V. ; SETHI, R. ; ULLMAN, J. D.: *Compilers: principles, techniques, and tools*. Reading, Mass. [u.a.] : Addison-Wesley, 2002 (36. Auflage). – ISBN 0–201–10088–6

[Alternate Computing Solutions 2005]
ALTERNATE COMPUTING SOLUTIONS: *JSVN*. 0.8. Alternate Computing Solutions, 2005. – Software, Abruf: 18. 9. 2005. http://jsvn.alternatecomputing.com/

[Barker 1992]
BARKER, R.: *Case method : Entity-Relationship-Modellierung*. Bonn [u.a.] : Addison-Wesley, 1992 (1. Auflage). – ISBN 3–89319–397–9

[Baumeister u. Petrowski 2002]
BAUMEISTER, J. ; PETROWSKI, T.: *VBA - Visual Basic for Applications: Programmieren unter Office XP*. München : Deutscher Taschenbuch-Verlag, 2002 (2. neu bearbeitete Auflage). – ISBN 3–423–50189–8

[Beer u. Firmenich 2003]
BEER, D. G. ; FIRMENICH, B.: Freigabestände von strukturierten Objektversionsmengen in Bauprojekten. In: HEMPEL, L. (Hrsg.) ; GÜRLEBECK, G. (Hrsg.) ; KÖNKE, C. (Hrsg.): *Internationales Kolloquium über Anwendungen der Informatik und Mathematik in Architektur und Bauwesen (IKM)*. Weimar : Bauhaus-Universität Weimar, Juni 2003. – ISSN 1611–4086. – http://e-pub.uni-weimar.de/volltexte/2004/9/

[Beer 2002]
BEER, Daniel G.: *Objektzugriff und -manipulation in* AUTOCAD. Weimar, Bauhaus-Universität Weimar, Seminarunterlagen, 2002. – http://www.uni-weimar.de/~bauinf/lehre/CAEPlan/SCAEPlan.html

[Berczuk u. Appleton 2002]
BERCZUK, S. P. ; APPLETON, B.: *Software configuration management patterns: effective teamwork and practical integration*. Boston, [Mass.] [u.a.] : Addison-Wesley, 2002. – ISBN 0–201–74117–2

[Beucke 2000]
BEUCKE, K.: Objektorientierte Konzepte in der Tragwerksplanung. In: KATZ, C. (Hrsg.): *Software für Statik und Konstruktion*, Balkema, 2000. – ISBN 90–5809–137–6

[Beucke 2002]
BEUCKE, K.: *CAE im Planungsprozess*. Weimar, Bauhaus-Universität Weimar, Vorlesungsskript, 2002. – http://www.uni-weimar.de/~bauinf/lehre/CAEPlan/CAEPlan.html

[Beucke u. Beer 2005]
BEUCKE, K. ; BEER, D. G.: Net Distributed Applications in Civil Engineering: Approach and Transition Concept for CAD Systems. In: SOIBELMAN, L. (Hrsg.) ; PENA-MORA, F. (Hrsg.): *Digital Proceedings of the International Conference on Computing in Civil Engineering (ICCC 2005)*, American Society of Civil Engineers (ASCE), Juli 2005. – ISBN 0–7844–0794–0

[Beucke u. Freundt 2000]
BEUCKE, K. ; FREUNDT, M.: *Grundlagen grafischer Nutzeroberflächen*. Weimar, Bauhaus-Universität Weimar, Vorlesungsskript, 2000. – http://www.uni-weimar.de/~bauinf/lehre/SWTIng/SWTIng.html

[Beucke u. Heinrich 2002]
Beucke, K. ; Heinrich, T.: *Kommunikationssysteme*. Weimar, Bauhaus-Universität Weimar, Vorlesungsskript, 2002. – http://www.uni-weimar.de/~bauinf/lehre/Bi_3/Bi_3.html

[Beucke u. a. 2002]
Beucke, K. E. ; Beer, D. G. ; Freundt, M. ; Heinrich, T. ; Huhnt, W.: *Sonderforschungsbereich SFB 524, Teilprojekt D1 (Phase 1): Strukturierung von Informationen und Prozessen für Revitalisierungsaufgaben*. November 2002. – Webseite. http://www.uni-weimar.de/sfb/last-website.1st/d1.html

[Beucke u. a. 2004a]
Beucke, K. E. ; Beer, D. G. ; Freundt, M. ; Heinrich, T. ; Kirschke, H.: *Sonderforschungsbereich SFB 524, Teilprojekt D1 (Phase 2): Methoden und Werkzeuge zur Planung und Steuerung von Bauprozessen*. November 2004. – Webseite. http://www.uni-weimar.de/sfb/d1.html

[Beucke u. a. 2004b]
Beucke, K. E. ; Donath, D. ; Hübler, R. ; Werner, F.: *Sonderforschungsbereich SFB 524, Projektbereich D: Informationsverarbeitung und Kommunikation*. November 2004. – Webseite. http://www.uni-weimar.de/sfb/gesamt.html#Struktur

[Bierman 2000]
Bierman, G. M.: *Using XML as an Object Interchange Format*. Mai 2000. – Webseite, Abruf: 15. 9. 2005. http://www.odmg.org/oifml.pdf

[Bilchuk 2005]
Bilchuk, I.: *Generalisierte Informationsstrukturen für Anwendungen im Bauwesen*. Berlin, Technische Universität Berlin, Dissertation, 2005

[Bilchuk u. Pahl 2004]
Bilchuk, I. ; Pahl, P. J.: Entwicklung einer Datenbasis für Anwendungen im Bauwesen. In: Beucke, K. (Hrsg.) ; Firmenich, B. (Hrsg.) ; Donath, D. (Hrsg.) ; Fruchter, R. (Hrsg.) ; Roddis, K. (Hrsg.): *Xth International Conference on Computing in Civil and Building Engineering*. Weimar : Bauhaus-Universität Weimar, Juni 2004. – ISBN 3–86068–213–X. – http://e-pub.uni-weimar.de/volltexte/2004/169/

[Blandy u. a. 2002]
Blandy, J. ; Fogel, K. ; Collins-Sussman, B.: *Subversion – A New Version Control System*. 2002. – Webseite, Abruf: 12. 05. 2004. http://subversion.tigris.org/files/documents/15/18/svn-design.pdf

[Booch 1993]
Booch, G.: *Object oriented design with applications*. Redwood City, Calif. [u.a.] : Benjamin/Cummings, 1993. – ISBN 0–8053–0091–0

[Booch 1995]
Booch, G.: *Objektorientierte Analyse und Design mit praktischen Anwendungsbeispielen*. Bonn [u.a.] : Addison-Wesley, 1995. – ISBN 3–89319–673–0

[Booch u. a. 2005]
Booch, G. ; Rumbaugh, J. ; Jacobson, I.: *The unified modeling language user guide*. Upper Saddle River : Addison-Wesley, 2005 (2. Auflage). – ISBN 0–321–26797–4

[Bretschneider 1998]
BRETSCHNEIDER, D.: *Modellierung rechnerunterstützter, kooperativer Arbeit in der Tragwerksplanung*. Düsseldorf : VDI, 1998 (Fortschrittsberichte VDI Reihe 4 Nr. 151). – ISBN 3–18–315104–9

[Broy 1998]
BROY, M.: *Systemstrukturen und Theoretische Informatik*. Springer-Lehrbuch. Berlin [u.a.] : Springer, 1998 (2. überarbeitete Auflage). – ISBN 3–540–64392–3

[Bucher u. Müller 2004]
BUCHER, C. ; MÜLLER, K.-H.: *Sonderforschungsbereich SFB 524: Werkstoffe und Konstruktionen für die Revitalisierung von Bauwerken*. November 2004. – Webseite. http://www.uni-weimar.de/sfb/

[Budszuhn 2005]
BUDSZUHN, F.: *Subversion: Einführung in das neue Versionsmanagement-System*. Galileo Computing. Bonn : Galileo Press, 2005 (1. Auflage). – ISBN 3–89842–603–3

[Carnegie Mellon Software Engineering Institute 2005]
CARNEGIE MELLON SOFTWARE ENGINEERING INSTITUTE: *How Do You Define Software Architecture?* August 2005. – Webseite, Abruf: 19. 08. 2005. http://www.sei.cmu.edu/architecture/definitions.html

[Cattell 2000]
CATTELL, R. G. G.: *The object data standard: ODMG 3.0*. The Morgan Kaufman series in data management systems. San Francisco [u.a.] : Morgan Kaufmann, 2000. – ISBN 1–558–60647–5

[Cederqvist 2002]
CEDERQVIST, P.: *Version management with CVS*. Bristol : Network Theory, 2002. – ISBN 0–9541617–1–8

[Claus u. Schwill 2003]
CLAUS, V. ; SCHWILL, A.: *Duden, Informatik: ein Fachlexikon für Studium und Praxis*. Mannheim, Leipzig, Wien, Zürich : Dudenverlag, 2003. – ISBN 3–411–10023–0

[Collins-Sussman u. a. 2005]
COLLINS-SUSSMAN, B. ; FITZPATRICK, B. W. ; PILATO, C. M.: *Versionskontrolle mit Subversion: Software-Projekte intelligent koordinieren*. Beijing [u.a.] : O'Reilly, 2005 (1. Auflage). – http://svnbook.red-bean.com/. – ISBN 3–89721–390–7

[Date 2000]
DATE, C. J.: *An introduction to database systems*. Addison-Wesley world student series. Reading, Mass. [u.a.] : Addison-Wesley-Longman, 2000. – ISBN 0–201–68419–5

[Dehnhardt 1999]
DEHNHARDT, W.: *Anwendungsprogrammierung mit JDBC*. München : Hanser, 1999. – ISBN 3–446–21265–5

[DIN 6789-5 1995]
Norm DIN 6789-5 Oktober 1995. *Dokumentationssystematik: Freigabe in der Technischen Produktdokumentation*

[DIN 6789-7 2003]
Norm DIN 6789-7 November 2003. *Dokumentationssystematik: Qualitätskriterien für Freigabeprozesse digitaler Produktdaten*

[Downing 1998]
DOWNING, T. B.: *Java RMI: remote method invocation*. Foster City, Calif. [u.a.] : IDG Books Worldwide, 1998. – ISBN 0–7645–8043–4

[Eckel 1998]
ECKEL, B.: *Thinking in Java*. Prentice Hall PTR : Upper Saddle River, NJ, 1998. – ISBN 0–13–659723–8

[Elmasri u. Navathe 2000]
ELMASRI, R. ; NAVATHE, S. B.: *Fundamentals of database systems*. Reading, Mass. [u.a.] : Addison-Wesley, 2000 (3. überarbeitete Auflage). – ISBN 0–201–54263–3

[Emmerich 2003]
EMMERICH, W.: *Konstruktion von verteilten Objekten*. dpunkt-Lehrbuch. Heidelberg : dpunkt, 2003. – ISBN 3–89864–140–6

[Firmenich 2000]
FIRMENICH, B.: Eine CAD-Systemarchitektur zur Unterstützung der verteilten Bearbeitung im Internet. In: HANFF, J. (Hrsg.) ; KASPAREK, E. (Hrsg.) ; RUESS, M. (Hrsg.) ; SCHUTTE, G. (Hrsg.): *Forum Bauinformatik 2000*. Düsseldorf : VDI, August 2000. – ISBN 3–18–316304–3. – Beitrag nicht im Tagungsband erschienen.

[Firmenich 2001]
FIRMENICH, B.: *CAD im Bauplanungsprozess: Verteilte Bearbeitung einer strukturierten Menge von Objektversionen*. Weimar, Bauhaus-Universität Weimar, Dissertation, 2001. – [Firmenich 2002a]

[Firmenich 2002a]
FIRMENICH, B.: *CAD im Bauplanungsprozess: Verteilte Bearbeitung einer strukturierten Menge von Objektversionen*. Aachen : Shaker, 2002 (Berichte aus dem Bauwesen). – *http://www.shaker.de/Online-Gesamtkatalog/Details.idc?ISBN=3-8265-9924-1*. – ISBN 3–8265–9924–1

[Firmenich 2002b]
FIRMENICH, B.: Operations for the distributed synchronous cooperation of a shared versioned data model in the planning process. In: HSIEH, S.-H. (Hrsg.) ; LEU, L.-J. (Hrsg.) ; CHEN, C.-S. (Hrsg.) ; LI, J.-F. (Hrsg.): *Proceedings of the 9th International Conference on Computing in Civil and Building Engineering (ICCCBE-IX)* Bd. 2. Taipei : Department of Civil Engineering, National Taiwan University, April 2002. – ISBN 986–80222–0–7, S. 1081–1086

[Firmenich 2004a]
FIRMENICH, B.: *DFG-Schwerpunktprogramm 1103, Arbeitsgruppe Verteilte Produktmodelle*. November 2004. – Webseite. *http://www.iib.bauing.tu-darmstadt.de/dfg-spp1103/de/arbeitsgruppen/Produktmodell.html*

[Firmenich 2004b]
FIRMENICH, B.: *Eine formale Sprache für operative CAD-Modelle im Bauwesen*. August 2004. – Webseite, Abruf: 20. 07. 2005. *http://www.uni-weimar.de/~bauinf/cad/forschung/DFG/opCAD/opCAD.html*

[Firmenich 2004c]
FIRMENICH, B.: A Novel Modelling Approach for the Exchange of CAD Information in Civil Engineering. In: DIKBAS, A. (Hrsg.) ; SCHERER, R. (Hrsg.): *Proceedings of the*

5th European Conference on Product and Process Modelling in the Building and related Industries : eWork and eBusiness in Architecture, Engineering and Construction. Leiden [u.a.] : Balkema, September 2004. – ISBN 04–1535–938–4, S. 77–83

[Firmenich 2004d]
FIRMENICH, B.: Product Models in Network Based Co-operation in Structural Engineering. In: BEUCKE, K. (Hrsg.) ; FIRMENICH, B. (Hrsg.) ; DONATH, D. (Hrsg.) ; FRUCHTER, R. (Hrsg.) ; RODDIS, K. (Hrsg.): *Xth International Conference on Computing in Civil and Building Engineering*. Weimar : Bauhaus-Universität Weimar, Juni 2004. – ISBN 3–86068–213–X. – *http://e-pub.uni-weimar.de/volltexte/2004/217/*

[Firmenich 2005]
FIRMENICH, B.: *CAD in der Bauinformatik I*. Weimar, Bauhaus-Universität Weimar, Vorlesungsskript, 2005. – *http://www.uni-weimar.de/~bauinf/cad*

[Firmenich 2006]
FIRMENICH, B.: **CAD**EMIA. Januar 2006. – Webseite. *http://www.cademia.org*

[Firmenich u. a. 2004]
FIRMENICH, B. ; BEER, D. G. ; RICHTER, T.: *Projekt „interCAD“: Entwurf und Verifizierung einer CAD-Systemarchitektur zur Unterstützung der verteilten technischen Bearbeitung im Konstruktiven Ingenieurbau*. November 2004. – Webseite. *http://www.uni-weimar.de/~bauinf/forschung/DFG/interCAD/interCAD.html*

[Firmenich u. a. 2005]
FIRMENICH, B. ; KOCH, C. ; RICHTER, T. ; BEER, D. G.: Versioning structured object sets using text based Version Control Systems. In: SCHERER, R. J. (Hrsg.) ; KATRANUSCHKOV, P. (Hrsg.) ; SCHAPKE, S.-E. (Hrsg.): *CIB-W78 – 22nd Conference on Information Technology in Construction*. Dresden : Institute for Construction Informatics, TU Dresden, Juli 2005. – ISBN 3–86005–478–3, S. 105–112

[Fish 2005]
FISH, S.: *Version Control System Comparison*. Mai 2005. – Webseite, Abruf: 01. 06. 2005. *http://better-scm.berlios.de/comparison/comparison.html*

[Flanagan 2003]
FLANAGAN, D.: *Java in a Nutshell*. Beijing [u.a.] : O'Reilly, 2003 (Deutsche Ausgabe, 4. Auflage). – ISBN 3–89721–332–X

[Fogel u. Bar 2002]
FOGEL, K. ; BAR, M.: *Open Source-Projekte mit CVS: Verteilte Softwareentwicklung mit dem Concurrent Versions System*. Bonn : mitp, 2002 (2. erweiterte und überarbeitete Auflage). – ISBN 3–8266–0816–X

[Foley u. a. 2002]
FOLEY, J. D. ; DAM, A. van ; FEINER, S. ; HUGHES, J. ; PHILLIPS, R.: *Introduction to Computer Graphics*. Boston [u.a.] : Addison-Wesley, 2002 (14. Auflage). – ISBN 0–201–60921–5

[Forman u. Forman 2004]
FORMAN, I. R. ; FORMAN, N.: *Java reflection in action*. Greenwich, Conn. : Manning, 2004. – ISBN 1–932394–18–4

[Fowler u. Scott 1999]
FOWLER, M. ; SCOTT, K.: *UML - konzentriert: Die Standardobjektmodellierungssprache anwenden*. Bonn [u.a.] : Addison-Wesley, 1999. – ISBN 3–8273–1329–5

[Friedl 2003]
FRIEDL, J.E.F.: *Reguläre Ausdrücke*. Beijing [u.a.] : O'Reilly, 2003 (Deutsche Ausgabe, 2. Auflage). – ISBN 3–89721–349–4

[Frischalowski u. Pallmer 2004]
FRISCHALOWSKI, D. ; PALLMER, J.: *CVS: Versionsverwaltung von Softwareprodukten*. Bildungsmedien IT-Training. Bodenheim : Herdt, 2004 (1. Auflage)

[Furix.com 2005]
FURIX.COM: *CompareDWG*. 2006. Furix AutoCAD Tools, 2005. – Software, Abruf: 25. 10. 2005. *http://www.furix.com/products/compare/index.htm*

[Gamma 2003]
GAMMA, E.: *Entwurfsmuster : Elemente wiederverwendbarer objektorientierter Software*. München [u.a.] : Addison-Wesley, 2003 (5. korrigierter Nachdruck). – Dt. Übersetzung von D. Riehle. – ISBN 3–8273–1862–9

[Gumm u. Sommer 2002]
GUMM, H.-P. ; SOMMER, M.: *Einführung in die Informatik*. München [u.a.] : R. Oldenbourg, 2002 (5. überarbeitete Auflage). – ISBN 3–486–25635–1

[Hanff 2003a]
HANFF, J.: *Abhängigkeiten zwischen Objekten in ingenieurwissenschaftlichen Anwendungen*. Berichte aus dem Bauwesen. Aachen : Shaker, 2003. – *http://www.shaker.de/Online-Gesamtkatalog/Details.idc?ISBN=3-8322-2280-4*. – ISBN 3–8322–2280–4

[Hanff 2003b]
HANFF, J.: *Abhängigkeiten zwischen Objekten in ingenieurwissenschaftlichen Anwendungen*. Berlin, Technische Universität Berlin, Dissertation, 2003. – [Hanff 2003a]

[Hartmann u. a. 2004]
HARTMANN, D. ; MEISSNER, U. F. ; RÜPPEL, U. ; BILEK, J. ; THEISS, M.: Integration of Product Model Databases in Multi-Agent-Systems. In: BEUCKE, K. (Hrsg.) ; FIRMENICH, B. (Hrsg.) ; DONATH, D. (Hrsg.) ; FRUCHTER, R. (Hrsg.) ; RODDIS, K. (Hrsg.): *Xth International Conference on Computing in Civil and Building Engineering*. Weimar : Bauhaus-Universität Weimar, Juni 2004. – ISBN 3–86068–213–X. – *http://e-pub.uni-weimar.de/volltexte/2004/145/*

[Hass 2003]
HASS, A. M. J.: *Configuration management principles and practice*. The Agile software development series. Boston [u.a.] : Addison-Wesley, 2003. – ISBN 0–321–11766–2

[Hauser u. Löwer 2004]
HAUSER, T. ; LÖWER, U. M.: *Web-Services: Die Standards*. Galileo computing. Bonn : Galileo Press, 2004 (1. Auflage). – ISBN 3–89842–393–X

[Heinrich 2005]
HEINRICH, T.: Schnittstelle zwischen Produkt und Prozess – Navigation im Planungsprozesss. In: BUCHER, C. (Hrsg.): *Revitalisierung von Bauwerken: Veröffentlichungen des Sonderforschungsbereiches 524 „Werkstoffe und Konstruktionen für die Revitalisierung von Bauwerken“*. Weimar : Universitätsverlag Weimar, Februar 2005 (Schriften der Bauhaus-Universität 117). – ISBN 386068–248–2, S. 119–122

[Heinrich 2006]
HEINRICH, T.: *Schnittstelle zwischen Produkt und Prozess – Navigation im Planungsprozesss (Arbeitstitel)*. Weimar, Bauhaus-Universität Weimar, Dissertation (in Bearbeitung), 2006

[Herold u. Meyer 1995]
HEROLD, H. ; MEYER, M.: *SCCS und RCS: Versionsverwaltung und UNIX*. UNIX und seine Werkzeuge. Bonn [u.a.] : Addison-Wesley, 1995 (1. Auflage). – ISBN 3–89319–754–0

[Heuer u. Saake 2000]
HEUER, A. ; SAAKE, G.: *Datenbanken: Konzepte und Sprachen*. Bonn : mitp, 2000 (2. aktualisierte und erweiterte Auflage). – ISBN 3–8266–0619–1

[Hopcroft u. a. 2002]
HOPCROFT, J. E. ; MOTWANI, R. ; ULLMAN, J. D.: *Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie*. Informatik: Theoretische Informatik. München : Pearson Studium, 2002 (2. überarbeitete Auflage). – ISBN 3–8273–7020–5

[Horstmann u. Cornell 2003]
HORSTMANN, C. S. ; CORNELL, G.: *Core Java 2 - Expertenwissen*. München : Markt + Technik, 2003. – ISBN 3–8272–6228–3

[Huhn 2003a]
HUHN, M.: CAD by XML. Was XML in Planungssystemen leisten kann. In: HEMPEL, L. (Hrsg.) ; GÜRLEBECK, G. (Hrsg.) ; KÖNKE, C. (Hrsg.): *Internationales Kolloquium über Anwendungen der Informatik und Mathematik in Architektur und Bauwesen (IKM)*. Weimar : Bauhaus-Universität Weimar, Juni 2003. – ISSN 1611–4086. – http://e-pub.uni-weimar.de/volltexte/2005/327/

[Huhn 2003b]
HUHN, M.: Enabling cooperative design tasks - A collaboration platform by using Web Services. In: CHA, J. (Hrsg.) ; JARDIM-GONÇALVES, R. (Hrsg.): *Advanced design, production and management systems*. Lisse [u.a.] : Balkema, 2003. – ISBN 90–5809–623–8

[Hunt u. McIlroy 1976]
HUNT, J. W. ; MCILROY, M. D.: An Algorithm for Differential File Comparison. In: *Computing Science Technical Report* (1976), Nr. 41, Abruf: 17. 05. 2004. – http://www.cs.dartmouth.edu/~doug/

[IAI 2004]
IAI: *Industry Foundation Classes (IFC)*. Juli 2004. – Webseite, Abruf: 21. 07. 2005. http://www.iai-international.org/Model/IFC(ifcXML)Specs.html

[Jordan u. Russel 2003]
JORDAN, D. ; RUSSEL, C.: *Java data objects*. Beijing [u.a.] : O'Reilly, 2003. – ISBN 0–596–00276–9

[Kazakos u. a. 2002]
KAZAKOS, W. ; SCHMIDT, A. ; TOMCZYK, P.: *Datenbanken und XML : Konzepte, Anwendungen, Systeme*. Xpert.press. Berlin [u.a.] : Springer, 2002. – ISBN 3–540–41956–X

[Kemper u. Eickler 2004]
KEMPER, A. ; EICKLER, A.: *Datenbanksysteme: Eine Einführung*. München [u.a.] : Oldenbourg, 2004 (5. aktualisierte und erweiterte Auflage). – ISBN 3–486–27392–2

[Kim 1991]
KIM, W.: *Introduction to object-oriented databases*. Computer systems series. Cambridge, Mass. [u.a.] : MIT Press, 1991 (2. Auflage). – ISBN 0–262–11124–1

[Kitz 2004]
KITZ, A.: *IT-Projektmanagement*. Bonn : Galileo Press, 2004 (1. Auflage). – ISBN 3–898–42396–4

[Klauer 2005]
KLAUER, T.: *Eine prozessorientierte Kooperationsplattform für Bauprojekte auf Basis eines internetbasierten Workflow-Managements*. Berichte des Instituts für Numerische Methoden und Informatik im Bauwesen. Aachen : Shaker, 2005. – *http://www.shaker.de/Online-Gesamtkatalog/Details.idc?ISBN=3-8322-4091-8*. – ISBN 3–8322–4091–8

[Klettke u. Meyer 2003]
KLETTKE, M. ; MEYER, H.: *XML & Datenbanken: Konzepte, Sprachen und Systeme*. xml.bibliothek. Heidelberg : dpunkt, 2003 (1. Auflage). – ISBN 3–89864–148–1

[Klingelhöller 2001]
KLINGELHÖLLER, H.: *Dokumentenmanagementsysteme: Handbuch zur Einführung*. Xpert.press. Berlin [u.a.] : Springer, 2001. – ISBN 3–540–41250–6

[Koch 2005]
KOCH, C.: Standardisierung von CAD durch eine Sprache für operative CAD-Modelle. In: SCHLEY, F. (Hrsg.) ; WEBER, L. (Hrsg.): *Forum Bauinformatik 2005*. Cottbus : Lehrstuhl Bauinformatik, BTU, September 2005 (Progress in Bauinformatik). – ISBN 3–934934–11–0, S. 26–33

[Krüger 2002]
KRÜGER, G.: *Handbuch der Java-Programmierung*. Bonn [u.a.] : Addison-Wesley, 2002. – ISBN 3–8273–1949–8

[Laabs 1998]
LAABS, A.: *Methoden für die Modellierung mit Objekten im Bauingenieurwesen*. Berichte aus dem Bauwesen. Aachen : Shaker, 1998. – *http://www.shaker.de/Online-Gesamtkatalog/Details.idc?ISBN=3-8265-3888-9*. – ISBN 3–8265–3888–9

[Langner 2002]
LANGNER, T.: *Verteilte Anwendungen mit Java: Enterprise-Architekturen im Web mit CORBA, XML/SOAP, JSP, (E)JB und JDBC*. New technology. München : Markt und Technik, 2002. – ISBN 3–8272–6296–8

[Lausen 2005]
LAUSEN, G.: *Datenbanken: Grundlagen und XML-Technologien*. Hochschultaschenbuch. München : Elsevier, Spektrum, 2005 (1. Auflage). – ISBN 3–8274–1488–1

[Lin 1996]
LIN, Y.-L.: *Configuration Management in Terms of Logical Structures*. Providence, Rhode Island, Brown University, Ph.D. Dissertation, 1996. – *http://www.cs.brown.edu/publications/techreports/reports/CS-95-31.html*

[Loney u. Theriault 2001]
LONEY, K. ; THERIAULT, M.: *Oracle 8i DBA-Handbuch*. The authorized Oracle Press editions. München [u.a.] : Hanser, 2001. – ISBN 3–446–21569–7

[MacKenzie u. a. 2003]
MacKenzie, D. ; Eggert, P. ; Stallmann, R.: *Comparing and Merging Files with GNU Diff and Patch*. Bristol : Network Theory, 2003 (2. Auflage). – http://www.gnu.org/software/diffutils/manual/ps/diff.ps.gz. – ISBN 0–9541617–5–0

[Matthiessen u. Unterstein 2003]
Matthiessen, G. ; Unterstein, M.: *Relationale Datenbanken und SQL – Konzepte der Entwicklung und Anwendung*. München [u.a.] : Addison-Wesley, 2003 (3. aktualisierte Auflage). – ISBN 3–8273–2085–2

[McAuley 2000]
McAuley, C.: *Programming AutoCAD 2000 using ObjectARX*. Albany[u.a.] : Autodesk Press, 2000. – ISBN 0–7668–0643–x

[Meißner u. Rüppel 1999]
Meissner, U. F. ; Rüppel, U.: *DFG-Schwerpunktprogramm 1103: Vernetzt-kooperative Planungsprozesse im Konstruktiven Ingenieurbau*. Februar 1999. – Webseite, Abruf: 01. 06. 2005. http://www.dfg-spp1103.de/

[Meißner u. Rüppel 2003]
Meissner, U. F. ; Rüppel, U.: Vernetzt-kooperative Planung mit Computern - Grundlagen und Methoden der Bauinformatik. In: Hempel, L. (Hrsg.) ; Gürlebeck, G. (Hrsg.) ; Könke, C. (Hrsg.): *Internationales Kolloquium über Anwendungen der Informatik und Mathematik in Architektur und Bauwesen (IKM)*. Weimar : Bauhaus-Universität Weimar, Juni 2003. – ISSN 1611–4086. – http://e-pub.uni-weimar.de/volltexte/2004/15/

[Meißner u. a. 2000]
Meissner, U. F. ; Rüppel, U. ; Petersen, M.: Co-operative Management of Product Models in the Planning Process of Structural Engineering. In: Atluri, S. N. (Hrsg.) ; Brust, F. W. (Hrsg.): *Advances in Computational Engineering & Sciences* Bd. 1. Forsyth/ USA : Tech Science Press, 2000. – ISBN 0–9657001–3–5, S. 240–245

[Meißner u. a. 2005]
Meissner, U. F. ; Rüppel, U. ; Theiss, M. ; Lange, M.: An Agent-based Model-Compound for Fire Protection Engineering. In: Soibelman, L. (Hrsg.) ; Pena-Mora, F. (Hrsg.): *Digital Proceedings of the International Conference on Computing in Civil Engineering (ICCC 2005)*, American Society of Civil Engineers (ASCE), Juli 2005. – ISBN 0–7844–0794–0

[Meyer 1990]
Meyer, B.: *Objektorientierte Softwareentwicklung*. München [u.a.] : Hanser [u.a.], 1990. – Dt. Übersetzung von W. Simonsmeier. – ISBN 3–446–15773–5

[Müller u. a. 2003]
Müller, J. ; Kirschke, H. ; Köcher, J. ; Petter, S.: Untersuchung und Erarbeitung eines Leitfadens zum Dokumentenmanagement in KMU des Bauwesens. In: Hempel, L. (Hrsg.) ; Gürlebeck, G. (Hrsg.) ; Könke, C. (Hrsg.): *Internationales Kolloquium über Anwendungen der Informatik und Mathematik in Architektur und Bauwesen (IKM)*. Weimar : Bauhaus-Universität Weimar, Juni 2003. – ISSN 1611–4086. – http://e-pub.uni-weimar.de/volltexte/2005/348/

[Myers 1986]
Myers, E.: An O(ND) Difference Algorithm and Its Variations. In: *Algorithmica* (1986), Nr. 1, S. 251–266, Abruf: 17. 05. 2004. – http://www.xmailserver.org/diff2.pdf

[Neufein 2003]
NEUFEIN, S.: Vielseitig erweiterbar – Versions-Kontrolle mit Subversion. In: *PHPmagazin* (2003), Juni, Abruf: 12. 05. 2004. – *http://www2.php-mag.de/itr/online_artikel/psecom,id,431,nodeid,62.html*

[Nour 2006]
NOUR, M. M.: *A flexible model for incorporating construction product data into building information models (Arbeitstitel).* Weimar, Bauhaus-Universität Weimar, Dissertation (in Bearbeitung), 2006

[Nussbaumer u. Mistelbacher 2003]
NUSSBAUMER, A. ; MISTELBACHER, A.: *XML ge-packt.* Ge-packte Referenz. Bonn : mitp, 2003 (1. Auflage). – ISBN 3–8266–0690–6

[Object Database Management Group 2005]
OBJECT DATABASE MANAGEMENT GROUP: *Object Data Management Group – The Standard for Storing Objects.* 2005. – Webseite, Abruf: 15. 9. 2005. *http://www.odmg.org/*

[Olbrich 1998]
OLBRICH, M.: *Relationenorientiertes Modellieren mit Objekten in der Bauinformatik.* Düsseldorf : VDI, 1998. – ISBN 3–18–315004–2

[Pahl u. Beucke 2000]
PAHL, P. J. ; BEUCKE, K.: Neuere Konzepte des CAD im Bauwesen; Stand und Entwicklungen. In: KIRSCHKE, H. (Hrsg.): *Internationales Kolloquium über Anwendungen der Informatik und Mathematik in Architektur und Bauwesen (IKM).* Weimar : Bauhaus-Universität Weimar, 2000. – *http://e-pub.uni-weimar.de/volltexte/2005/590/*

[Pahl u. Damrath 2000]
PAHL, P. J. ; DAMRATH, R.: *Mathematische Grundlagen der Ingenieurinformatik.* Berlin [u.a.] : Springer, 2000. – ISBN 3–540–60501–0

[Purdy 2001]
PURDY, G. N.: *CVS kurz und gut.* Beijing [u.a.] : O'Reilly, 2001 (1. Auflage). – ISBN 3–89721–229–3

[Ramunno 2005]
RAMUNNO, E. G.: *Vergleich und Zusammenführung unterschiedlicher Planungsstände.* Weimar, Bauhaus-Universität Weimar, Bachelorarbeit, November 2005. – *http://www.uni-weimar.de/~bauinf/Publikationen/BA/BARamunno.pdf*

[Reinhard u. Soeder 2001]
REINHARD, F. ; SOEDER, H.: *dtv-Atlas Mathematik.* München : Deutscher Taschenbuch-Verlag, 2001 (12. Auflage Band 1). – ISBN 3–423–03007–0

[Richter 2003]
RICHTER, T.: Ein Java-Paket zur Verarbeitung von Datenstrukturen in beliebigen Datenquellen. In: KAAPKE, K. (Hrsg.) ; WULF, A. (Hrsg.): *15. Forum Bauinformatik.* Aachen : Shaker, Oktober 2003. – ISBN 3–8322–2022–4, S. 136–146

[Richter 2006]
RICHTER, T.: *Ein Konzept für eine ingenieurgerechte Benutzerschnittstelle für versionierte Objektmodelle (Arbeitstitel).* Weimar, Bauhaus-Universität Weimar, Dissertation (in Bearbeitung), 2006

[Rüppel u. Meißner 2000]
RÜPPEL, U. ; MEISSNER, U. F.: Co-operative Structural Engineering in Distributed Systems. In: FRUCHTER, R. (Hrsg.) ; PENA-MORA, F. (Hrsg.) ; RODDIS, W. M. K. (Hrsg.): *Proceedings of the 8th International Conference on Computing in Civil & Building Engineering (ICCCBE-VIII)* Bd. 1. Stanford : ASCE, August 2000. – ISBN 0–7844–0513–1, S. 504–509

[Rüppel u. Meißner 2003]
RÜPPEL, U. ; MEISSNER, U. F.: Vernetzt-kooperative Planungsprozesse im Konstruktiven Ingenieurbau. In: NAGL, M. (Hrsg.) ; B.WESTFECHTEL (Hrsg.): *Modelle, Werkzeuge und Infrastrukturen zur Unterstützung von Entwicklungsprozessen.* Weinheim : Wiley-VCH, 2003. – ISBN 3–527–27769–2, S. 117–136

[Rüppel u. a. 2002]
RÜPPEL, U. ; MEISSNER, U. F. ; THEISS, M.: An Agent-Based Platform for Collaborative Building Engineering. In: HSIEH, S.-H. (Hrsg.) ; LEU, L.-J. (Hrsg.) ; CHEN, C.-S. (Hrsg.) ; LI, J.-F. (Hrsg.): *Proceedings of the 9th International Conference on Computing in Civil and Building Engineering (ICCCBE-IX)* Bd. 2. Taipei : Department of Civil Engineering, National Taiwan University, April 2002. – ISBN 986–80222–0–7, S. 1053–1056

[Ruh u. a. 2001]
RUH, W. R. ; MAGINNIS, F. X. ; BROWN, W. J.: *Enterprise Application Integration.* Wiley tech brief series. New York [u.a.] : Wiley, 2001. – ISBN 0–471–37641–8

[Rumbaugh u. a. 1991]
RUMBAUGH, J. ; BLAHA, M. ; PREMERLANI, W. ; EDDY, F. ; LORENSEN, W.: *Object-oriented modeling and design.* Englewood Cliffs, New Jersey : Prentice Hall [u.a.], 1991 (Prentice-Hall international editions). – ISBN 0–13–630054–5

[Saake u. a. 1997]
SAAKE, G. ; SCHMITT, I. ; TÜRKER, C.: *Objektdatenbanken: Konzepte, Sprachen, Architekturen.* Informatik Lehrbuch-Reihe. Bonn [u.a.] : Internat. Thomson Publ., 1997 (1. Auflage). – ISBN 3–8266–0258–7

[Scheid 2000]
SCHEID, H.: *Duden Rechnen und Mathematik.* Mannheim [u.a.] : Dudenverlag, 2000 (6. überarbeitete Auflage). – ISBN 3–411–05346–1

[Schicker 2000]
SCHICKER, E.: *Datenbanken und SQL: Eine praxisorientierte Einführung mit Hinweisen zu Oracle und MS-Access.* Informatik & Praxis. Stuttgart [u.a.] : Teubner, 2000 (3. durchgesehene Auflage). – ISBN 3–519–22991–9

[Schröder 1990]
SCHRÖDER, K.: *SCCS: Versionshaltung unter Unix.* Unix-Palette. München [u.a.] : Oldenbourg, 1990. – ISBN 3–486–21525–6

[Sleepycat 2005]
SLEEPYCAT: *Berkeley database.* 2005. – Software, Abruf: 22. 07. 2005. http://www.sleepycat.com/

[Smolka 1992]
SMOLKA, G.: Feature Constraint Logics for Unification Grammars. In: *Journal of Logic Programming* 12 (1992), Nr. 1, 2, S. 51–87. – http://www.ps.uni-sb.de/Papers/abstracts/FeatureConstraintLog.html

[Stroustrup 2002]
STROUSTRUP, B.: *Die C++-Programmiersprache*. Programmers choise. München [u.a.] : Addison-Wesley, 2002 (4. Auflage). – ISBN 3–8273–1660–X

[Stubblebine 2004]
STUBBLEBINE, T.: *Reguläre Ausdrücke - kurz & gut*. O'Reillys Taschenbibliothek. Beijing [u.a.] : O'Reilly, 2004 (Deutsche Ausgabe, 1. Auflage). – ISBN 3–89721–264–1

[Sun Microsystems, Inc. 2003a]
SUN MICROSYSTEMS, INC.: JAVA-*API*. 1.4.2. Sun Microsystems, Inc., 2003. – Software, Abruf: 20. 11. 2005. *http://java.sun.com/j2se/1.4.2/docs/api/index.html*

[Sun Microsystems, Inc. 2003b]
SUN MICROSYSTEMS, INC.: *javacc Project home*. 3.2. August 2003. – Software, Abruf: 10. 6. 2005. *https://javacc.dev.java.net/*

[Tanenbaum 2003]
TANENBAUM, A. S.: *Verteilte Systeme: Grundlagen und Paradigmen*. München [u.a.] : Pearson Studium, 2003. – ISBN 3–8273–7057–4

[Theiß 2005]
THEISS, M.: *Agentenbasierter Modellverbund am Beispiel des baulichen Brandschutzes in der Gebäudeplanung*. Berichte des Instituts für Numerische Methoden und Informatik im Bauwesen. Aachen : Shaker, 2005. – *http://www.shaker.de/Online-Gesamtkatalog/Details.idc?ISBN=3-8322-4359-3*. – ISBN 3–8322–4359–3

[Theiß u. Bilek 2003]
THEISS, M. ; BILEK, J.: *Agenten im Bauwesen*. 2003. – Webseite, Abruf: 6. 11. 2005. *http://www.agenten-im-bauwesen.net/*

[tmate.org 2005]
TMATE.ORG: *JavaSVN – The only pure Java Subversion library in the world!* 1.0.0. TMate Software, November 2005. – Software, Abruf: 18. 9. 2005. *http://tmate.org/svn/*

[Turau 1996]
TURAU, V.: *Algorithmische Graphentheorie*. Bonn [u.a.] : Addison-Wesley, 1996. – ISBN 3–89319–938–1

[Ullenboom 2003]
ULLENBOOM, C.: *Java ist auch eine Insel*. Bonn : Galileo Computing, 2003 (3. Auflage). – ISBN 3–89842–365–4

[van der Vlist 2003]
VAN DER VLIST, E.: *XML Schema: XML-Daten modellieren*. Köln [u.a.] : O'Reilly, 2003 (1. Auflage, Deutsche Ausgabe). – ISBN 3–89721–345–1

[Vesperman 2004]
VESPERMAN, J.: *CVS: Versionskontrolle und Quellcode-Management*. Köln [u.a.] : O'Reilly, 2004 (Deutsche Ausgabe, 1. Auflage). – ISBN 3–89721–369–9

[W3C 2005]
W3C: *XQuery 1.0: An XML Query Language*. April 2005. – Webseite, Abruf: 13. 06. 2005. *http://www.w3.org/TR/xquery/*

[Wagenknecht 2003]
WAGENKNECHT, C.: *Algorithmen und Komplexität*. Informatik interaktiv. München [u.a.] : Carl Hanser, 2003. – ISBN 3–446–22314–2

[Weiß u. Jakob 2005]
WEISS, G. ; JAKOB, R.: *Agentenorientierte Softwareentwicklung*. Xpert.press. Berlin [u.a.] : Springer, 2005. – ISBN 3–540–00062–3

[Weiß 2004a]
WEISS, J.: *Vereinfachte Integration von Fachwissen in Computerprogramme am Beispiel eines Planungssystems für die Bauindustrie*. München, Universität der Bundeswehr München, Dissertation, 2004

[Weiß 2004b]
WEISS, J.: *Vereinfachte Integration von Fachwissen in Computerprogramme am Beispiel eines Planungssystems für die Bauindustrie*. Neubiberg : Universität der Bundeswehr München, 2004 (Berichte aus dem konstruktiven Ingenieurbau 04/01). – *http://nbn-resolving.de/urn/resolver.pl?urn=nbn:de:bvb:706-773*. – ISBN 1–932394–18–4

[Weise u. a. 2004]
WEISE, M. ; KATRANUSCHKOV, P. ; SCHERER, R. J.: Managing Long Transactions in Model Server Based Collaboration. In: DIKBAS, A. (Hrsg.) ; SCHERER, R. (Hrsg.): *Proceedings of the 5th European Conference on Product and Process Modelling in the Building and related Industries: eWork and eBusiness in Architecture, Engineering and Construction*. Leiden [u.a.] : Balkema, September 2004. – ISBN 04–1535–938–4, S. 187–194

[WeltWeitBau 2005]
WELTWEITBAU: *PlanDiffViewer*. 2005. – Software, Abruf: 13. 06. 2005. *http://www.weltweitbau.de/de/PlanNet/PlanDiffViewer/PlanDiffViewer.html*

[Willenbacher 2002]
WILLENBACHER, H.: *Interaktive verknüpfungsbasierte Bauwerksmodellierung als Integrationsplattform für den Bauwerkslebenszyklus*. Weimar, Bauhaus Universität Weimar, Dissertation, 2002. – *http://e-pub.uni-weimar.de/volltexte/2004/32/*

[Zeller 1997]
ZELLER, A.: *Configuration Management with Version Sets - A Unified Software Versioning Model and its Applications*. Braunschweig, Technische Universität Braunschweig, Dissertation, 1997

A.9 Index

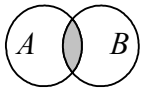
Anhang B

Fachliche Grundlagen

B.1 Mengen- und Relationenalgebra

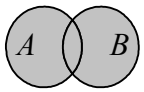
[Pahl u. Damrath 2000], [Reinhard u. Soeder 2001], [Scheid 2000]

Schnittmenge: Die *Schnittmenge* $A \cap B$ zweier Mengen A und B enthält alle Elemente x, die in beiden Mengen A und B enthalten sind. Für Relationen wird der Schnittmengenoperator $\sqcap$ verwendet.



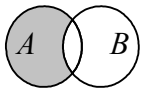
$$A \cap B = \{x \mid x \in A \wedge x \in B\} \tag{B.1}$$

Vereinigungsmenge: Die *Vereinigungsmenge* $A \cup B$ zweier Mengen A und B enthält alle Elemente x, die entweder in der Menge A oder in der Menge B enthalten sind. Für Relationen wird der Vereinigungsmengenoperator $\sqcup$ verwendet.



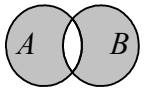
$$A \cup B = \{x \mid x \in A \vee x \in B\} \tag{B.2}$$

Differenzmenge: Die *Differenzmenge* $A \setminus B$ zweier Mengen A und B enthält alle Elemente x, die in der Menge A, jedoch nicht in der Menge B enthalten sind.

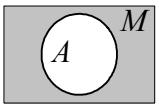


$$A \setminus B = \{x \mid x \in A \wedge x \notin B\} \tag{B.3}$$

Symmetrische Differenzmenge: Die *symmetrische Differenzmenge* $A \oplus B$ zweier Mengen A und B enthält alle Elemente x, die entweder in der Menge A, oder in der Menge B, jedoch nicht in beiden enthalten sind.

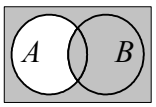


$$A \oplus B = \{x \mid x \in A \oplus x \in B\} = (A \cup B) \setminus (A \cap B) \tag{B.4}$$



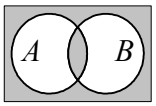
Komplementmenge: Die *Komplementmenge* $\overline{A}$ zu einer Menge A bezüglich einer Menge M, die Obermenge von A ist ($A \subseteq M$), enthält alle Elemente von M, die nicht in in A enthalten sind.

$$\overline{A} = M \setminus A \tag{B.5}$$



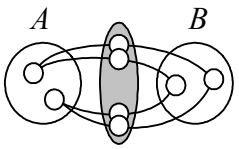
Implikationsmenge: Die *Implikationsmenge* $A \Rightarrow B$ enthält alle Elemente, die, wenn sie in A enthalten sind, dann auch in B enthalten sein müssen.

$$A \Rightarrow B = \overline{A} \cup B \tag{B.6}$$



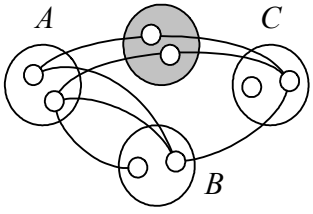
Äquivalenzmenge: Die *Äquivalenzmenge* $A \Leftrightarrow B$ enthält alle Elemente, die entweder in A und B enthalten sind oder in keiner der beiden Mengen.

$$A \Leftrightarrow B = (\overline{A} \cap \overline{B}) \cup (A \cap B) \tag{B.7}$$



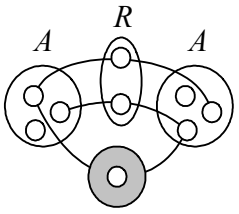
Kartesisches Produkt: Das *kartesische Produkt* $A \times B$ zweier Mengen A und B verknüpft alle Elemente der Menge A mit allen Elementen der Menge B. Es entsteht eine binäre Relation, die geordnete Paare aus Elementen der Menge A und B enthält.

$$A \times B = \{(a, b) \mid a \in A \wedge b \in B\} \tag{B.8}$$



Kettung: Die *Kettung* $\boldsymbol{R} \circ \boldsymbol{S}$ zweier Relationen, die auf den Mengen A und B ($\boldsymbol{R} \sqsubseteq A \times B$) bzw. B und C ($\boldsymbol{S} \sqsubseteq B \times C$) basieren, ist die Verknüpfung beider Relationen über ein gemeinsames Element der gemeinsamen Menge B. Sie ist eine Teilmenge des kartesischen Produktes $A \times C$.

$$\boldsymbol{R} \circ \boldsymbol{S} = \{(a, c) \in A \times C \mid \underset{b \in B}{\forall} (aRb \wedge bSc)\} \tag{B.9}$$



Transitive Hülle: Die *transitive Hülle* $<\boldsymbol{R}>_t$ einer Relation $\boldsymbol{R}$ erweitert diese um Paare (x, y), für die es einen Weg $\langle x, \ldots, y \rangle$ von x nach y in $\boldsymbol{R}$gibt. Sie lässt sich durch mehrfache Kettung berechnen.

$$<\boldsymbol{R}>_t = \{(x, y) \mid \text{Es existiert ein Weg } \langle x, \ldots, y \rangle \text{ in } \boldsymbol{R}\} = \boldsymbol{R}^m \tag{B.10}$$

B.2 Sprache

[Broy 1998], [Gumm u. Sommer 2002], [Hopcroft u. a. 2002], [Claus u. Schwill 2003]

Zeichen: Ein *Zeichen* z ist ein Element aus einer zur Darstellung von Informationen vereinbarten endlichen Menge Z der Zeichen.

Alphabet: Ein *Alphabet* A ist eine Teilmenge der Menge der Zeichen Z.

$$A = \{z \mid z \in Z \text{ ist ein Zeichen des Alphabets } A\} \tag{B.11}$$

Wort: Ein *Wort* w^n der Länge $n > 0$ ist eine Folge $\langle z_i \rangle_n$ von Zeichen aus einem Alphabet A. Ein leeres Wort w^n der Länge $n = 0$ enthält kein Zeichen.

$$w^n = \langle z_0, z_1, \ldots, z_{n-1} \rangle \in A^n \tag{B.12}$$
$$w^0 = \{\epsilon\} \in A^0 \tag{B.13}$$

Wortschatz: Der *Wortschatz* A^* eines Alphabets A ist die Menge aller endlichen Wörter w der Länge $l \leq n$, die durch Aneinanderreihung nicht notwendigerweise verschiedener Zeichen z_i des Alphabets A entsteht.

$$A^* = \bigcup_{n \in \mathbb{N}} A^n \tag{B.14}$$

Syntax: Die *Syntax* (*grch. „syntaxis“, dt. „Zusammenordnung“*) X über einem Alphabet A ist eine Teilmenge $X \subseteq A^*$ des Wortschatzes A^*. Sie definiert die Menge der Wörter, die für eine Sprache mit dieser Syntax sinnvoll ist.

$$X \subseteq A^* \tag{B.15}$$

Terminal, Symbol: Ein *Terminal* w_t bzw. *Symbol* ist ein spezielles Wort, das ein Konstrukt einer Sprache mit der Syntax X ist. Terminale werden in der Menge T zusammengefasst.

$$T = \{w_t \in X \mid w_t \text{ ist ein Konstrukt einer Sprache der Syntax } X\} \tag{B.16}$$

Nichtterminal: Ein *Nichtterminal* w_n ist ein Wort, das kein Konstrukt einer Sprache mit der Syntax X darstellt. Nichtterminale werden in der Menge N zusammengefasst. Ein Wort kann entweder Terminal oder ein Nichtterminal sein.

$$N = \{w_n \in X \mid w_n \text{ ist kein Konstrukt einer Sprache der Syntax } X\} = X \setminus T \tag{B.17}$$
$$X = N \cup T, N \cap T = \emptyset \tag{B.18}$$

Regulärer Ausdruck: Terminale[1] werden mit Hilfe von *regulären Ausdrücken* formuliert. Ein regulärer Ausdruck (*engl. „regular expression“*) $regExpr$ über einem Alphabet A beschreibt eine reguläre Menge Reg von Zeichenfolgen. Folgende Ausdrücke sind reguläre Ausdrücke über dem Alphabet A:

1. Ein Zeichen z, das sich durch folgende Operationen aus der regulären Menge Reg auswählen lässt,

.	*Beliebiges* Zeichen
$[ab]$	Zeichen a *oder* b
$[a-z]$	*Alle* Zeichen von a bis z
$[\hat{}a]$	*Nicht* das Zeichen a

2. eine Zeichenfolge $\langle z_i \rangle \in A^*$,
3. eine Zeichenfolge $\langle z_i \rangle$, die sich durch folgende Operationen aus der regulären Menge Reg ergibt.

$z_i \vert z_k$ oder $[z_i, z_k]$	*Entweder* z_i *oder* z_k
$z_i?$	z_i *Entweder* 0 oder 1 mal
z_i+	*Mindestens* 1 mal z_i
z_i*	*Beliebig* oft z_i
$z_i\{n, m\}$	*Mindestens* n mal, *maximal* m mal z_i

Produktion, Regel: Eine *Produktion* oder *Regel* r ist ein Paar $p \rightarrow q \in X \times X$ von Wörtern p und q, das die Ersetzung von Nichtterminalen durch Terminale definiert. Dazu muss das Wort p mindestens ein Nichtterminal enthalten.

$$r = p \rightarrow q \in (N \cup T)^* \cdot N \cdot (N \cup T)^* \times (N \cup T)^* \subseteq X \times X \tag{B.19}$$

Produktionssystem, Regelsystem: Die Menge R aller Regeln r wird *Produktionssystem* oder *Regelsystem* genannt.

$$R = \{r \mid r \text{ ist eine Regel}\} \tag{B.20}$$

Grammatik: Eine *Grammatik* G ist ein Viertupel (N, T, R, S) einer Menge N von Nichtterminalen, eine Menge T von Terminalen, einer Menge R von Regeln und einem Startsymbol S, das Element von N ist.

$$G = (N, T, R, S) \qquad S \in N \tag{B.21}$$

Sprache: Eine (formale) *Sprache* (*engl. „language“*) $L(G)$ ist die mit der Grammatik G beschriebene Menge aller Wörter, die sich aus dem Startsymbol S ableiten und nur aus Terminalen bestehen.

$$L(G) = \{w \in T^* \mid S \xrightarrow{*} w \text{ bezüglich } G\} \tag{B.22}$$

[1] [Friedl 2003], [Stubblebine 2004]

B.3 Entwurfsmuster

[Gamma 2003]

B.3.1 Beobachter

Komponenten:

- `Subjekt:` Das *`Subjekt`* ist ein beobachtbares Objekt mit einer Menge von Beobachtern. Es bietet Methoden zur An- und Abmeldung (`meldeAn()`, `meldeAb()`) sowie zur Benachrichtigung von Beobachtern (`benachrichtige()`).

- `KonkretesSubjekt:` Das *konkrete Subjekt* ist ein konkretes, beobachtbares Objekt mit einem Zustand. Es benachrichtigt alle Beobachter bei Zustandsänderungen und kann seinen Zustand zurückliefern (`gibZustand()`).

- `Beobachter:` Der *`Beobachter`* definiert die Aktualisierungsmethode `aktualisiere()`.

- `KonkreterBeobachter:` Ein *`KonkreterBeobachter`* beobachtet ein Subjekt. Er implementiert die Aktualisierungsmethode unter Verwendung der Methoden des Subjekts.

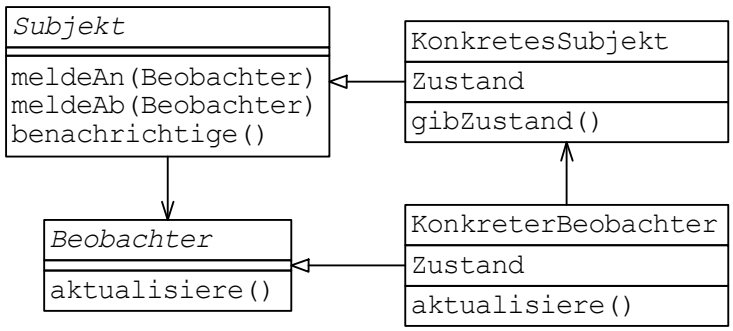


Abbildung B.1: Entwurfsmuster Beobachter

Anwendung:

- Reaktion abhängiger Objekte auf Änderungen an Objekten

- Benachrichtigung von Objekten, deren Struktur unbekannt ist

B.3.2 Kommando

Komponenten:

- `Befehl:` Die Klasse *`Befehl`* ist die Basisklasse aller Befehle. Sie definiert die Schnittstelle zum Ausführen des Befehls (`doCmd()`), zum Rückgängigmachen (`undo()`) und zum erneuten Ausführen nach dem Rückgängigmachen (`redo()`).

- `KonkreterBefehl:` Die Klasse *`KonkreterBefehl`* speichert den zum Ausführen nötigen `Zustand`, darunter typischerweise auch einen Verweis auf den Empfänger, und implementiert die Befehlsschnittstelle.

- `Auslöser:` Der *`Auslöser`* erzeugt einen konkreten Befehl und versieht ihn mit einem Verweis auf den Empfänger und allen anderen nötigen Informationen.

- `Aufrufer:` Der *`Aufrufer`* führt Befehle (erneut) aus oder macht sie rückgängig.

- `Empfänger:` Der konkrete Befehl ruft Methoden des *Empfänger*objektes auf, um seine Aktion auszuführen. An den Empfänger werden keine besonderen Anforderungen gestellt. Er muss nichts über die anderen Akteure wissen. Somit kann jede Klasse als Empfänger dienen.

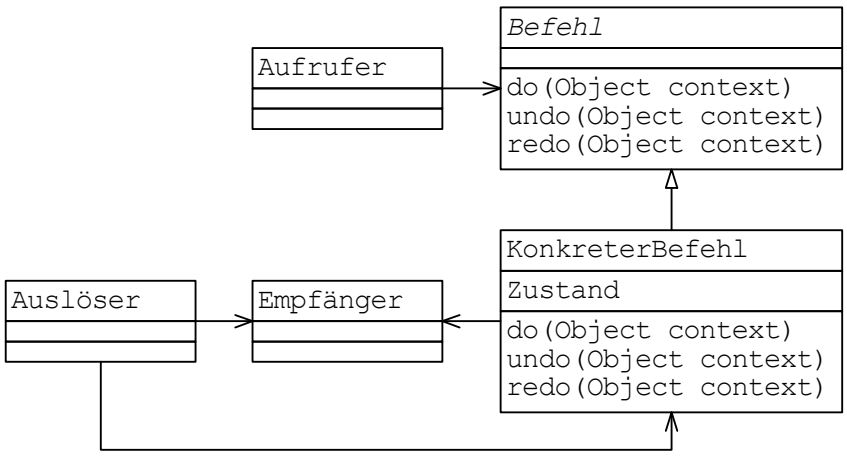


Abbildung B.2: Entwurfsmuster Kommando

Anwendung:

- Ausführung von parametrisierten Aktionen

- Erstellen und Ausführen von Aktionen zu verschiedenen Zeiten oder in anderem Kontext (Thread, Prozess, Rechner)

- Implementation eines undo-/ redo-Mechanismus

B.3.3 Stellvertreter

Komponenten:

- `Proxy`: Die Klasse ***`Proxy`*** (*dt. „Stellvertreter“*) verwaltet die Referenz auf ein zu vertretendes Objekt und kontrolliert den Zugriff auf dieses. Beide haben die gleiche Schnittstelle.
- `Subjekt`: Die Klasse ***`Subject`*** ist die gemeinsame Schnittstelle von Stellvertreter und Objekt.

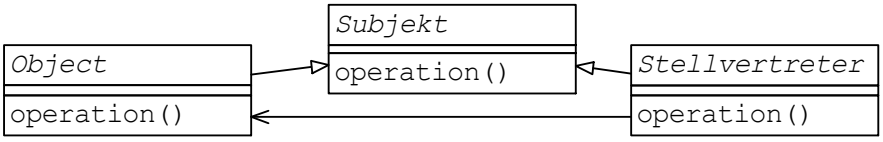


Abbildung B.3: Entwurfsmuster Stellvertreter

Anwendung:

- **Remote-Proxy:** Lokaler Stellvertreter für ein Objekt in einem anderen (entfernten, *engl. „remote“*) Adressraum
- **Virtual Proxy:** Verzögerung von Operationen mit hohem Ressourcenbedarf
- **Schutz-Proxy:** Einführung von Zugriffsrechten
- **Smart Reference:** Ausführen weiterer Operationen beim Zugriff auf das Objekt

B.3.4 Iterator

Komponenten:

- `Iterator`: Der ***`Iterator`*** definiert die Schnittstelle zum Zugriff auf die Elemente (`gibElement()`) und zum Traversieren des Aggregates (`weiter()`)
- `KonkreterIterator`: Der ***`KonkreteIterator`*** speichert die Position im Aggregat.
- `Aggregat`: Das ***`Aggregat`*** definiert die Schnittstelle zum Erzeugen des Iterators.
- `KonkretesAggregat`: Das ***`KonkreteAggregat`*** erzeugt einen konkreten Iterator.

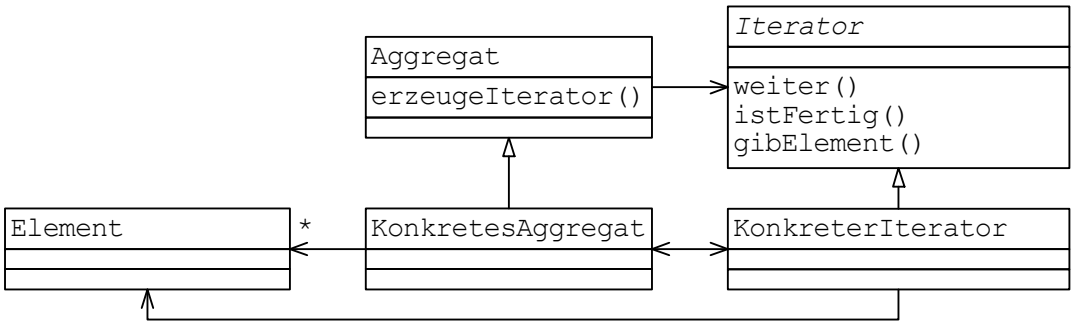


Abbildung B.4: Entwurfsmuster Iterator

Anwendung:

- Kapselung der Repräsentation der Elemente im Aggregat
- Kapselung der Traversierung der Elemente im Aggregat

B.4 Komplexität

[Hopcroft u. a. 2002], [Wagenknecht 2003]

Komplexität: Die *Komplexität* $g(n)$ eines Problems ist die (minimale) Anzahl an Rechenschritten, die ein Algorithmus zur Lösung eines Problems benötigt. Sie wird in Abhängigkeit von der Länge n der Eingabe angegeben und für große n asymptotisch unter Verwendung des Landau-Symbols $O()$ für die obere Schranke abgeschätzt.

$$O(f(n)) = \{g : \mathbb{N} \rightarrow \mathbb{N} \mid \underset{n_0>0}{\exists} \underset{c>0}{\exists} \underset{n>n_0}{\forall} g(n) \leq c \cdot f(n)\} \tag{B.23}$$

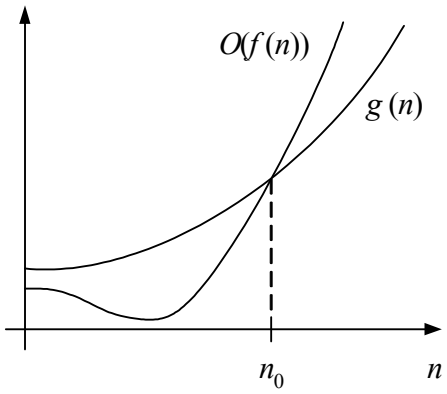


Abbildung B.5: Komplexität

Konstante Komplexität: Eine *konstante Komplexität* c wird mit $O(1)$ ausgedrückt.

$$c = O(1) \tag{B.24}$$

Worst case: Der schlechteste Fall (*engl. „worst case“*) gibt an, wie lange ein Algorithmus maximal braucht. Für viele Algorithmen gibt es nur wenige Eingaben, die die worst-case-Laufzeit erreichen, weshalb sie nicht unbedingt eine realistische Abschätzung ist.

Average case: Der durchschnittliche Fall (*engl. „average case“*) gibt die erwartete Laufzeit bei einer durchschnittlichen Zahl der Eingaben an.

Best case: Der beste Fall (*engl. „best case“*) gibt an, wie lange ein Algorithmus in jedem Fall braucht, also selbst für die Eingaben, auf denen er ideal arbeitet. Diese untere Schranke zur Lösung eines Problems wird nur recht selten angegeben, da sie nur für wenige Fälle zutrifft.

Komplexitätsregeln: Komplexitäten können über *Regeln* miteinander verknüpft werden. Die Addition von Komplexitäten entspricht der maximalen Komplexität der Summanden.

$$\sum_{i=1}^{m} O(f_i(n)) = O(max(f_i(n))) = max(O(f_i(n))) \tag{B.25}$$

Das Produkt von Komplexitäten entspricht der Komplexität des Produkts der Faktoren.

$$\prod_{i=1}^{m} O(f_i(n)) = O(\prod(f_i(n))) \tag{B.26}$$

Grundbausteine einer Programmiersprache: Um die Komplexität eines Algorithmus ermitteln zu können, bedarf es der Kenntnis der grundlegenden Bausteine der für die Umsetzung verwendeten Programmiersprache. Die Gesamtkomplexität des Algorithmus wird aus den Einzelkomplexitäten der Bestandteile mit Hilfe der genannten Komplexitätsregeln berechnet.

Anweisung: *Anweisungen* sind zum Beispiel Wertzuweisungen, Deklarationen, einfache Arithmetik $(+, -, \cdot, /)$, Vergleiche $(==,!=,<,>,<=,>=)$, Sprunganweisungen, Eingaben/Ausgaben usw. Die Laufzeit dafür ist zwar rechnerabhängig jedoch konstant $(O(1))$.

Sequenz: Eine *Sequenz* ist die aufeinander folgende Ausführung mehrerer Anweisungen. Dies entspricht der Addition von Komplexitäten[2].

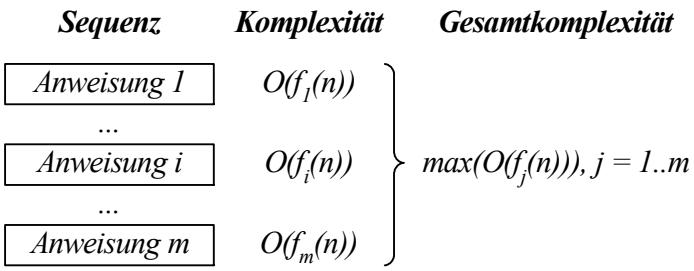


Abbildung B.6: Komplexität einer Sequenz

Schleife: Innerhalb einer *Schleife* wird ein Algorithmus der Komplexität $O(f(n_i))$, der abhängig vom Schleifendurchlauf i ist, m mal ausgeführt. Es ergibt sich:

$$O(f(n_1)) + \cdots + O(f(n_i)) + \cdots + O(f(n_m)) = max(O(f(n_i))) \tag{B.27}$$

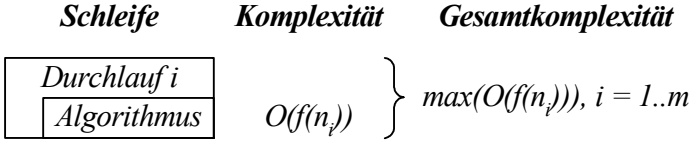


Abbildung B.7: Komplexität einer Schleife

[2] Siehe Formel (B.25) auf Seite 193.

Verzweigung: Bei einer *Verzweigung* mit m Möglichkeiten wird je nach Erfüllung einer Bedingung ein Algorithmus A_i mit der Komplexität $O(f_i(n))$ ausgeführt. Die Ausführung des i-ten Algorithmus habe die Wahrscheinlichkeit p_i mit $\sum_{i=1}^{m} p_i = 1$. Im worst case (best case) wird der Algorithmus mit der größten (geringsten) Komplexität gewählt. Im Normalfall ist Komplexität für die über die Ausführungswahrscheinlichkeit gewichtete Summe zu bilden.

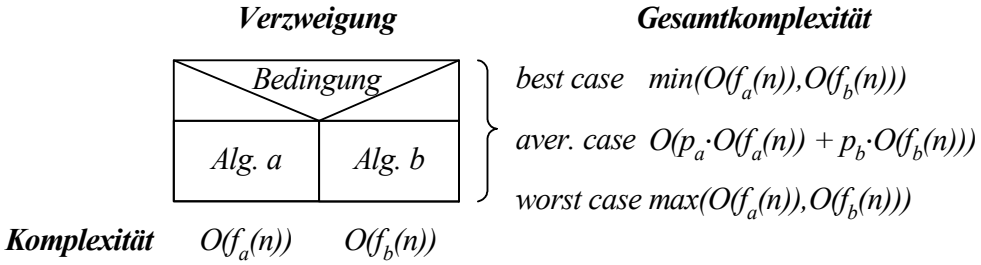


Abbildung B.8: Komplexität einer Verzweigung

Algorithmen: Die Komplexität von *Algorithmen*, die nicht im Quellcode vorliegen, lässt sich nicht ableiten. Sie muss anhand von Tests empirisch ermittelt werden. Für diese Tests ist eine Menge von Messwerten und eine Anzahl von Wiederholungen der Messungen festzulegen.

Aus der Menge der Messwerte kann auf eine Komplexität geschlossen werden, indem ein Kurvenverlauf angenommen wird und mit Hilfe einer Regression die Kurvenparameter so bestimmt werden, dass der Abstand zwischen den Messwerten und der Kurve minimal wird. Der Regressionskoeffizient

$$R^2 = \frac{\sum_n^{i=1}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_n^{i=1}(x_i - \overline{x})^2 \cdot \sum_n^{i=1}(y_i - \overline{y})}} \quad \text{mit} \quad \overline{x} = \frac{1}{n} \cdot \sum_n^{i=1} x_i, \overline{y} = \frac{1}{n} \cdot \sum_n^{i=1} y_i \tag{B.28}$$

bestätigt die Annahme oder widerlegt sie: Ist der Wert nahe 1, so ist der Zusammenhang signifikant, ist er 0, so besteht kein Zusammenhang und es ist ein anderer Kurvenverlauf anzunehmen.

Die Anzahl der Wiederholungen gibt Auskunft über die Verteilung der Messwerte. Diese ist im Allgemeinen keine Normalverteilung, da die erste Ausführung eines Algorithmus oft mehr Zeit benötigt als ein weiterer Durchlauf, weil Informationen zwischengespeichert werden (Cache). Außerdem können parallel laufende Programme im Hintergrund die Messwerte verfälschen. Es ist jedoch zu beachten, dass die Regression abhängig von der Art der Verteilung ist.

Anhang C

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Bei der Auswahl und Auswertung von Material haben mir andere Personen weder entgeltlich noch unentgeltlich geholfen.

Weitere Personen waren an der inhaltlich-materiellen Erstellung der vorliegenden Arbeit nicht beteiligt. Insbesondere habe ich hierfür nicht die entgeltliche Hilfe von Vermittlungs- bzw. Beratungsdiensten (Promotionsberater oder anderer Personen) in Anspruch genommen. Niemand hat von mir unmittelbar oder mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt.

Ich versichere an Eides statt, dass ich nach bestem Gewissen die reine Wahrheit gesagt und nichts verschwiegen habe.

Vor Aufnahme der obigen Versicherung an Eides statt wurde ich über die Bedeutung der eidesstattlichen Versicherung und die strafrechtlichen Folgen einer unrichtigen oder unvollständigen eidesstattlichen Versicherung belehrt.

Weimar, den 31. März 2006 Daniel G. Beer

Anhang D

Über den Autor

D.1 Lebenslauf

Allgemeine Angaben

28.02.1976	Geboren in Gera/ Thüringen
Familienstand	Ledig, keine Kinder

Ausbildung

09/82 – 08/90	Polytechnische Oberschule „Werner Lamberz“, St. Gangloff/ Thüringen
09/90 – 08/94	Zabel-Gymnasium Gera/ Thüringen, Abitur (1,0) und Zabelpreis
09/94 – 07/00	Bauingenieur-Studium, Bauhaus-Universität Weimar, Diplom (1,7)
10/95 – 09/96	Zivildienst, Pflegeheim Münchenbernsdorf/ Thüringen

Beruflicher Werdegang

04/98 - 09/98	Studentischer Mitarbeiter bei Prof. Dr.-Ing. K. Beucke, Lehrstuhl „Informatik im Bauwesen“, Bauhaus-Universität Weimar (Lehrtätigkeit)
08/00 – 03/06	Wissenschaftlicher Mitarbeiter bei Prof. Dr.-Ing. K. Beucke, Lehrstuhl „Informatik im Bauwesen“ (Projektbearbeitung und Lehre)
08/00 – 12/02	Mitarbeit im Sonderforschungsbereich SFB 524 - “Werkstoffe und Konstruktionen für die Revitalisierung von Bauwerken“
08/01 – 12/05	Durchführung von Lehrveranstaltungen (Grundlagen der Bauinformatik, CAE im Planungsprozess, Advanced Studies in Structural Engineering and CAE)
01/03 – 03/06	Bearbeitung des Teilprojekts „Entwurf und Verifizierung einer CAD-Systemarchitektur zur Unterstützung der verteilten technischen Bearbeitung im Konstruktiven Ingenieurbau“ im DFG-Schwerpunktprogramm 1103 „Vernetzt-kooperative Planungsprozesse im Konstruktiven Ingenieurbau“

D.2 Publikationen

[Beer u. a. 2001]
BEER, D. G. ; HEINRICH, T. ; HUHNT, W.: Methoden und Werkzeuge zur Planung der Planung. In: ROMBERG, R. (Hrsg.) ; SCHULZ, M. (Hrsg.): *13. Forum Bauinformatik*. Düsseldorf : VDI, Oktober 2001. – ISBN 3–18–316904–5, S. 154–161

[Beer 2002]
BEER, D. G.: Process Models as a Base for Communication in Revitalization Projects. In: TURK, Z. (Hrsg.) ; SCHERER, R. J. (Hrsg.): *Proceedings of the 4th European Conference on Product and Process Modelling in the Building and related Industries: eWork and eBusiness in Architecture, Engineering and Construction*. Leiden [u.a.] : Balkema, September 2002. – ISBN 90–5809–507, S. 231–236

[Beer 2003]
BEER, D. G.: Motivation für eine Sprache zur Handhabung strukturierter Objekversionsmengen. In: KAAPKE, K. (Hrsg.) ; WULF, A. (Hrsg.): *15. Forum Bauinformatik*. Aachen : Shaker, Oktober 2003. – ISBN 3–8322–3233–8, S. 128–145

[Beer u. Firmenich 2003]
BEER, D. G. ; FIRMENICH, B.: Freigabestände von strukturierten Objektversionsmengen in Bauprojekten. In: HEMPEL, L. (Hrsg.) ; GÜRLEBECK, G. (Hrsg.) ; KÖNKE, C. (Hrsg.): *Internationales Kolloquium über Anwendungen der Informatik und Mathematik in Architektur und Bauwesen (IKM)*. Weimar : Bauhaus-Universität Weimar, Juni 2003. – ISSN 1611–4086. – http://e-pub.uni-weimar.de/volltexte/2004/9/

[Beer u. a. 2004a]
BEER, D. G. ; FIRMENICH, B. ; RICHTER, T. ; BEUCKE, K.: A Concept for CAD Systems with Persistent Versioned Data Models. In: BEUCKE, K. (Hrsg.) ; FIRMENICH, B. (Hrsg.) ; DONATH, D. (Hrsg.) ; FRUCHTER, R. (Hrsg.) ; RODDIS, K. (Hrsg.): *Xth International Conference on Computing in Civil and Building Engineering*. Bauhaus-Universität Weimar. – ISBN 3–86068–213–X. – http://e-pub.uni-weimar.de/volltexte/2004/211/

[Beer u. a. 2004b]
BEER, D. G. ; FIRMENICH, B. ; RICHTER, T. ; BEUCKE, K.: A Persistence Interface for Versioned Object Models. In: DIKBAS, A. (Hrsg.) ; SCHERER, R. J. (Hrsg.): *Proceedings of the 5th European Conference on Product and Process Modelling in the Building and related Industries : eWork and eBusiness in Architecture, Engineering and Construction*. Leiden [u.a.] : Balkema, September 2004. – ISBN 04–1535–938–4, S. 49–57

[Beer u. a. 2006]
BEER, D. G. ; FIRMENICH, B. ; BEUCKE, K.: A system architecture for net-distributed applications in civil engineering. In: *Proceedings of the Joint International Conference on Computing and Decision Making in Civil and Building Engineering*. Juni 2006. – Paper accepted.

[Beucke u. Beer 2005]
BEUCKE, K. ; BEER, D. G.: Net Distributed Applications in Civil Engineering: Approach and Transition Concept for CAD Systems. In: SOIBELMAN, L. (Hrsg.) ; PENA-MORA, F. (Hrsg.): *Digital Proceedings of the International Conference on Computing in Civil Engineering (ICCC 2005)*, American Society of Civil Engineers (ASCE), Juli 2005. – ISBN 0–7844–0794–0

[Firmenich u. a. 2004]
FIRMENICH, B. ; BEER, D. G. ; RICHTER, T.: *Projekt „interCAD“: Entwurf und Verifizierung einer CAD-Systemarchitektur zur Unterstützung der verteilten technischen Bearbeitung im Konstruktiven Ingenieurbau*. November 2004. – Webseite. http://www.uni-weimar.de/~bauinf/forschung/DFG/interCAD/interCAD.html

[Firmenich u. a. 2005]
FIRMENICH, B. ; KOCH, C. ; RICHTER, T. ; BEER, D. G.: Versioning structured object sets using text based Version Control Systems. In: SCHERER, R. J. (Hrsg.) ; KATRANUSCHKOV, P. (Hrsg.) ; SCHAPKE, S.-E. (Hrsg.): *CIB-W78 – 22nd Conference on Information Technology in Construction*. Dresden : Institute for Construction Informatics, TU Dresden, Juli 2005. – ISBN 3–86005–478–3, S. 105–112

[Richter u. Beer 2004]
RICHTER, T. ; BEER, D. G.: Eine grafische Nutzerschnittstelle für versionierte Objektmodelle. In: *Forum Bauinformatik 2004*. Aachen : Shaker, September 2004. – ISBN 3–8322–3233–8, S. 264–271

[Vad u. a. 2002]
VAD, J. ; BEER, D. G. ; HEINRICH, T. ; HUHNT, W.: Werkzeuge zur Planung der Planung. In: BILEK, J. (Hrsg.): *14. Forum Bauinformatik*. Düsseldorf : VDI, September 2002. – ISBN 3–18–318104–5, S. 197–204

[Wahl u. a. 2003]
WAHL, A. ; FIRMENICH, B. ; BEER, D. G. ; BEUCKE, K.: JACIS - ACIS 3D Modellierung in Java. In: KAAPKE, K. ; WULF, A. (Hrsg.): *15. Forum Bauinformatik*. Aachen : Shaker, Oktober 2003. – ISBN 3–8322–2022–4, S. 275–285